Volume 24 Number 2

ACTA CYBERNETICA

Szeged, 2019

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). There are no page charges. An electronic version of the published paper is provided for the authors in PDF format.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements: title of the paper; author name(s) and affiliation; name, address and email of the corresponding author; an abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.).

When your paper is accepted for publication, you will be asked to upload the complete electronic version of your manuscript. For technical reasons we can only accept files in LaTeX format. It is advisable to prepare the manuscript following the guidelines described in the author kit available at `http://www.inf.u-szeged.hu/kutatas/acta-cybernetica/information-for-authors#AuthorKit` even at an early stage.

Submission and Review. Manuscripts must be submitted online using the editorial management system at `http://cyber.bibl.u-szeged.hu/index.php/actcybern/submission/wizard`. Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: `acta@inf.u-szeged.hu`

Web access. The above information along with the contents of past and current issues are available at the Acta Cybernetica homepage `https://www.inf.u-szeged.hu/en/kutatas/acta-cybernetica` .

Zoltan Kato
Department of Image Processing
and Computer Graphics
University of Szeged
Szeged, Hungary
kato@inf.u-szeged.hu

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Dana Petcu
Department of Computer Science
West University of Timisoara, Romania
petcu@info.uvt.ro

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
Heiko.Vogler@tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

Acta Cybernetica **24** (2019) 173–185.

An Elementary Proof of the General Poincaré Formula for λ-additive Measures*

József Dombi[a] and Tamás Jónás[b]

Abstract

In a previous paper of ours (see J. Dombi and T. Jónás. The general Poincaré formula for λ-additive measures. Information Sciences, 490:285-291, 2019.), we presented the general formula for λ-additive measure of union of n sets and gave a proof of it. That proof is based on the fact that the λ-additive measure is representable. In this study, a novel and elementary proof of the formula for λ-additive measure of the union of n sets is presented. Here, it is also demonstrated that, using elementary techniques, the well-known Poincaré formula of probability theory is just a limit case of our general formula.

Keywords: λ-additive measure, Poincaré formula

1 Introduction

Since the fuzzy measures (monotone measures) play an important role in describing various phenomena, over time there has been a steady interest in them (see, e.g. [13, 14, 22, 10, 8]). One of the most widely applied classes of monotone measures is the class of λ-additive measures (Sugeno λ-measures) (see, e.g. [21, 11, 12, 2, 1, 17]). [21]. Although there are many theoretical and applied articles that discuss the λ-additive measure, the general form of λ-additive measure of the union of n sets has just recently been identified [4]. In [4], we proved that if X is a finite set, $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \geq 2$, Q_λ is a λ-additive measure on X, $\lambda \in (-1, \infty)$ and

*This study was supported by the Hungarian Ministry of Human Capacities (grant 20391-3/2018/FEKUSTRAT).

[a]Department of Computer Algorithms and Artificial Intelligence, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary, E-mail: dombi@inf.u-szeged.hu

[b]Institute of Business Economics, Eötvös Loránd University, Szép utca 2., H-1053 Budapest, Hungary E-mail: jonas@gti.elte.hu

DOI: 10.14232/actacyb.24.2.2019.1

$\lambda \neq 0$, then

$$Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \\ = \frac{1}{\lambda}\left(\prod_{k=1}^{n}\left(\prod_{1\leq i_1<\cdots<i_k\leq n}\left(1+\lambda Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right)\right)\right)^{(-1)^{k-1}} - 1\right), \tag{1}$$

where $\mathcal{P}(X)$ denotes the power set of X. Our proof in [4] is based on the fact that Q_λ is representable [2]; that is, one has $Q_\lambda = h_\lambda \circ \mu$ for a uniquely determined additive measure $\mu : \mathcal{P}(X) \to [0,1]$, where $h_\lambda : [0,1] \to [0,1]$ is a strictly increasing bijection given via

$$h_\lambda(x) = \begin{cases} \dfrac{(1+\lambda)^x - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ x, & \text{if } \lambda = 0, \end{cases}$$

and $\lambda \in (-1, \infty)$. Here, we will prove the formula in Eq. (1) without utilizing the fact that Q_λ is representable. That is, we will give a novel and elementary proof of Eq. (1). Taking into account the fact that the fuzzy measures and the fuzzy measure related aggregation are important topics, it is worth mentioning that the formula in Eq. (1) may also be viewed as an aggregation related to the λ-additive measure, which is a fuzzy measure.

The well-known Poincaré formula of probability theory is

$$Pr\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{k=1}^{n}(-1)^{k-1}\sum_{1\leq i_1<\cdots<i_k\leq n} Pr\left(A_{i_1}\cap\cdots\cap A_{i_k}\right), \tag{2}$$

where Pr is a probability measure on X and $A_1, \ldots, A_n \in \mathcal{P}(X)$. Here, we will show that the Poincaré formula of probability theory given in Eq. (2) is a limit case of the general formula of λ-additive measure of the union of n sets given in Eq. (1). Namely, by using elementary techniques, we will prove that if X is a finite set, $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \geq 2$, Q_λ is a λ-additive measure on X and $\lambda \neq 0$, then

$$\lim_{\lambda\to 0} Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{k=1}^{n}(-1)^{k-1}\sum_{1\leq i_1<\cdots<i_k\leq n} Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right).$$

It is an acknowledged fact that the λ-additive measure is strongly connected with the belief- and plausibility measures of Dempster-Shafer theory (see, e.g. [22, 9, 5, 20, 7, 3, 16]), and with the theory of rough sets (see, e.g. [6, 24, 23, 15, 18, 19]). Hence, our formula for the λ-additive measure of the union of n sets may play an important role in these areas of computer science [4].

The rest of this paper is structured as follows. In Section 2, we will introduce our new result regarding the λ-additive measure of the union of n sets. Here, we will also prove that the Poincaré formula of probability theory is just a limit case

of our novel formula; that is, our formula may be viewed as the generalization of the Poincaré formula. Lastly, in Section 3, we will give a short summary of our findings and highlight our future research plans including the possible application of our results in network science.

In this study, we will use the common notations $\cap$ and $\cup$ for the intersection and union operations over sets, respectively. Also, will use the notation $\overline{A}$ for the complement of set A.

2 An elementary proof of the general Poincaré formula

Relaxing the additivity property of the probability measure, the following λ-additive measures were proposed by Sugeno in 1974 [21].

Definition 1. *The function $Q_\lambda : \mathcal{P}(X) \to [0,1]$ is a λ-additive measure (Sugeno λ-measure) on the finite set X, iff Q_λ satisfies the following requirements:*

(1) $Q_\lambda(X) = 1$

(2) for any $A, B \in \mathcal{P}(X)$ *and* $A \cap B = \emptyset$,

$$Q_\lambda(A \cup B) = Q_\lambda(A) + Q_\lambda(B) + \lambda Q_\lambda(A) Q_\lambda(B), \tag{3}$$

where $\lambda \in (-1, \infty)$ *and* $\mathcal{P}(X)$ *is the power set of* X.

Note that if X is an infinite set, then the continuity of function Q_λ is also required. From now on, $\mathcal{P}(X)$ will denote the power set of the finite set X and Q_λ will always denote a λ-additive measure on X.

The calculation of the λ-additive measure of the union of two disjoint sets is given in Definition 1. The following well-known lemma (see Theorem 4.6 (2) in [22]) shows how the λ-additive measure of the union of two sets can be computed when these sets are not necessarily disjoint.

Lemma 1. *If X is a finite set and Q_λ is a λ-additive measure on X, then for any $A, B \in \mathcal{P}(X)$,*

$$Q_\lambda(A \cup B) = \frac{Q_\lambda(A) + Q_\lambda(B) + \lambda Q_\lambda(A) Q_\lambda(B) - Q_\lambda(A \cap B)}{1 + \lambda Q_\lambda(A \cap B)}. \tag{4}$$

Proof. See the proof of Theorem 4.6 in [22]. □

Remark 1. Notice that if $\lambda = 0$, then Eq. (4) reduces to $Q_\lambda(A \cup B) = Q_\lambda(A) + Q_\lambda(B) - Q_\lambda(A \cap B)$, which has the same form as the probability measure of union of two sets.

Here, we will introduce a function and some quantities that we will utilize later on.

Definition 2. *The function* $p_{n,\lambda}^{(k)} : \mathcal{P}^n(X) \to \mathbb{R}$ *is given by*

$$p_{n,\lambda}^{(k)}(A_1, \ldots, A_n) = \prod_{1 \le i_1 < \cdots < i_k \le n} \left(1 + \lambda Q_\lambda \left(A_{i_1} \cap \cdots \cap A_{i_k}\right)\right),$$

where X *is a finite set,* $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \ge 2$, $1 \le k \le n$. *For the sake of simplicity, we will also use the notation* $Z_{n,\lambda}^{(k)} = p_{n,\lambda}^{(k)}(A_1, \ldots, A_n)$.

Later, we will utilize the following quantity to identify the general formula for the λ-additive measure of the union of n sets.

Definition 3. *The quantity* $Z_{n,\lambda}^{*(k)}$ *is given by*

$$Z_{n,\lambda}^{*(k)} = p_{n,\lambda}^{(k)}(A_1^*, \ldots, A_n^*),$$

where X *is a finite set,* $A_i^* = A_i \cap A_{n+1}$, $A_i, A_{n+1} \in \mathcal{P}(X)$, $n \ge 2$, $1 \le i \le n$, $1 \le k \le n$.

Here, we will demonstrate how the λ-additive measure of the union of n general sets can be computed. That is, we will discuss the Poincaré formula for the λ-additive measure. First, we will discuss some key properties of the quantities that we introduced previously.

Lemma 2. *If* X *is a finite set,* $A_1, \ldots, A_n, A_{n+1} \in \mathcal{P}(X)$, $A_i^* = A_i \cap A_{n+1}$ *and* $1 \le i \le n$, *then*

$$Z_{n,\lambda}^{*(k)} = p_{n,\lambda}^{(k)}(A_1^*, \ldots, A_n^*) =$$
$$= \prod_{1 \le i_1 < \cdots < i_k \le n} \left(1 + \lambda Q_\lambda \left(A_{i_1} \cap \cdots \cap A_{i_k} \cap A_{n+1}\right)\right),$$

where $n \ge 2$ *and* $1 \le k \le n$.

Proof. Exploiting the idempotent property of the set intersection operation, the lemma immediately follows from the definition of $Z_{n,\lambda}^{*(k)}$. □

The following lemma demonstrates a key connection between the $Z_{n,\lambda}^{*(k)}$ and $Z_{n,\lambda}^{(n)}$ quantities.

Lemma 3. *Let* X *be a finite set and let* $A_1, \ldots, A_n, A_{n+1} \in \mathcal{P}(X)$, $A_i^* = A_i \cap A_{n+1}$ *and* $1 \le i \le n$. *Then, for any* $n \ge 2$, $1 \le k \le n$, *the quantity* $Z_{n,\lambda}^{*(k)}$ *can be expressed in terms of* $Z_{n,\lambda}^{(k+1)}$ *and* $Z_{n+1,\lambda}^{(k+1)}$ *as follows:*

$$Z_{n,\lambda}^{*(k)} = \begin{cases} \dfrac{Z_{n+1,\lambda}^{(k+1)}}{Z_{n,\lambda}^{(k+1)}}, & \text{if } k < n \\ Z_{n+1,\lambda}^{(n+1)}, & \text{if } k = n. \end{cases} \tag{5}$$

Proof. Here, we will distinguish two cases: (1) $k < n$, (2) $k = n$.

(1) Based on Lemma 2, the following relation holds:

$$Z_{n,\lambda}^{*(k)} = p_{n,\lambda}^{(k)}(A_1^*, \ldots, A_n^*) = \\ = \prod_{1 \le i_1 < \cdots < i_k \le n} \left(1 + \lambda Q_\lambda \left(A_{i_1} \cap \cdots \cap A_{i_k} \cap A_{n+1}\right)\right). \tag{6}$$

Next, the right hand side of Eq. (6) can be written as

$$\prod_{1 \le i_1 < \cdots < i_k \le n} \left(1 + \lambda Q_\lambda \left(A_{i_1} \cap \cdots \cap A_{i_k} \cap A_{n+1}\right)\right) = \\ = \frac{\prod\limits_{1 \le i_1 < \cdots < i_{k+1} \le n+1} \left(1 + \lambda Q_\lambda \left(A_{i_1} \cap \cdots \cap A_{i_{k+1}}\right)\right)}{\prod\limits_{1 \le i_1 < \cdots < i_{k+1} \le n} \left(1 + \lambda Q_\lambda \left(A_{i_1} \cap \cdots \cap A_{i_{k+1}}\right)\right)} = \frac{Z_{n+1,\lambda}^{(k+1)}}{Z_{n,\lambda}^{(k+1)}}. \tag{7}$$

Notice that based on Definition 2, $Z_{n,\lambda}^{(k+1)}$ exists only if $k+1 \le n$; that is, if $k < n$. This explains why we need to differentiate the two cases in Eq. (5).

(2) If $k = n$, then based on Definition 3 and Definition 2,

$$Z_{n,\lambda}^{*(k)} = p_{n,\lambda}^{(k)}(A_1^*, \ldots, A_n^*) = 1 + \lambda Q_\lambda \left(A_1^* \cap \cdots \cap A_n^*\right) = \\ = 1 + \lambda Q_\lambda \left(\left(A_1 \cap A_{n+1}\right) \cap \cdots \cap \left(A_n \cap A_{n+1}\right)\right) = \\ = 1 + \lambda Q_\lambda \left(A_1 \cap \cdots \cap A_n \cap A_{n+1}\right) = p_{n+1,\lambda}^{(n+1)}(A_1, \ldots, A_{n+1}) = Z_{n+1,\lambda}^{(n+1)}. \tag{8}$$

□

The following example demonstrates the usefulness of Lemma 3. In this example, we will show how the quantity $Z_{3,\lambda}^{*(1)}$ can be expressed in terms of the quantities $Z_{4,\lambda}^{(2)}$ and $Z_{3,\lambda}^{(2)}$.

Example 1.

$$Z_{3,\lambda}^{*(1)} = \left(1 + \lambda Q_\lambda \left(A_1 \cap A_4\right)\right)\left(1 + \lambda Q_\lambda \left(A_2 \cap A_4\right)\right)\left(1 + \lambda Q_\lambda \left(A_3 \cap A_4\right)\right)$$

$$Z_{4,\lambda}^{(2)} = \left(1 + \lambda Q_\lambda \left(A_1 \cap A_2\right)\right)\left(1 + \lambda Q_\lambda \left(A_1 \cap A_3\right)\right)\left(1 + \lambda Q_\lambda \left(A_1 \cap A_4\right)\right) \cdot \\ \cdot \left(1 + \lambda Q_\lambda \left(A_2 \cap A_3\right)\right)\left(1 + \lambda Q_\lambda \left(A_2 \cap A_4\right)\right)\left(1 + \lambda Q_\lambda \left(A_3 \cap A_4\right)\right)$$

$$Z_{3,\lambda}^{(2)} = \left(1 + \lambda Q_\lambda \left(A_1 \cap A_2\right)\right)\left(1 + \lambda Q_\lambda \left(A_1 \cap A_3\right)\right)\left(1 + \lambda Q_\lambda \left(A_2 \cap A_3\right)\right)$$

It can be seen from the expressions of $Z_{3,\lambda}^{*(1)}$, $Z_{4,\lambda}^{(2)}$ and $Z_{3,\lambda}^{(2)}$ that the equation

$$Z_{3,\lambda}^{*(1)} = \frac{Z_{4,\lambda}^{(2)}}{Z_{3,\lambda}^{(2)}}$$

holds.

The next lemma shows how the λ-additive measure of set A_n can be expressed in terms of the $Z^{(1)}_{n,\lambda}$ and $Z^{(1)}_{n-1,\lambda}$ quantities.

Lemma 4. *If X is a finite set, $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \geq 3$ and $\lambda \neq 0$, then*

$$Q_\lambda(A_n) = \frac{1}{\lambda}\left(\frac{Z^{(1)}_{n,\lambda}}{Z^{(1)}_{n-1,\lambda}} - 1\right). \tag{9}$$

Proof. By utilizing the definitions of $Z^{(1)}_{n,\lambda}$ and $Z^{(1)}_{n-1,\lambda}$, we have

$$\frac{Z^{(1)}_{n,\lambda}}{Z^{(1)}_{n-1,\lambda}} = \frac{(1+\lambda Q_\lambda(A_1))\cdots(1+\lambda Q_\lambda(A_{n-1}))\,(1+\lambda Q_\lambda(A_n))}{(1+\lambda Q_\lambda(A_1))\cdots(1+\lambda Q_\lambda(A_{n-1}))} = \\ = 1+\lambda Q_\lambda(A_n),$$

from which Eq. (9) immediately follows. □

Now, we will state and prove a key theorem that allows us to compute the λ-additive measure of the union of n sets when the parameter λ is nonzero.

Theorem 1. *If X is a finite set, $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \geq 2$, Q_λ is a λ-additive measure on X and $\lambda \neq 0$, then*

$$Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \\ = \frac{1}{\lambda}\left(\prod_{k=1}^{n}\left(\prod_{1\leq i_1<\cdots<i_k\leq n}\left(1+\lambda Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right)\right)\right)^{(-1)^{k-1}} - 1\right). \tag{10}$$

Proof. By utilizing the definition of $Z^{(k)}_{n,\lambda}$, Eq. (10) can be written as

$$Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \frac{1}{\lambda}\left(\prod_{k=1}^{n}\left(Z^{(k)}_{n,\lambda}\right)^{(-1)^{k-1}} - 1\right). \tag{11}$$

It can be shown by direct calculation that the formula in Eq. (11) holds for $n = 2, 3$; that is,

$$Q_\lambda(A_1\cup A_2) = \frac{1}{\lambda}\left(\left(Z^{(1)}_{2,\lambda}\right)\left(Z^{(2)}_{2,\lambda}\right)^{-1} - 1\right)$$
$$Q_\lambda(A_1\cup A_2\cup A_3) = \frac{1}{\lambda}\left(\left(Z^{(1)}_{3,\lambda}\right)\left(Z^{(2)}_{3,\lambda}\right)^{-1}\left(Z^{(3)}_{3,\lambda}\right) - 1\right).$$

Here, we will apply induction; that is, we will prove that if Eq. (11) holds, then the equation

$$Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right) = \frac{1}{\lambda}\left(\prod_{k=1}^{n+1}\left(Z^{(k)}_{n+1,\lambda}\right)^{(-1)^{k-1}} - 1\right) \tag{12}$$

holds as well.

By making use of Lemma 1, the associativity of the set union operation and the distributivity of the set intersection operation over the set union operation, we have

$$\begin{aligned} Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right) &= Q_\lambda\left(\left(\bigcup_{i=1}^{n} A_i\right) \cup A_{n+1}\right) = \\ &= \frac{1}{1+\lambda Q_\lambda\left(\left(\bigcup_{i=1}^{n} A_i\right) \cap A_{n+1}\right)}\left(Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) + Q_\lambda(A_{n+1}) + \right. \\ &\left. +\lambda Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) Q_\lambda(A_{n+1}) - Q_\lambda\left(\left(\bigcup_{i=1}^{n} A_i\right) \cap A_{n+1}\right)\right) = \\ &= \frac{1}{1+\lambda Q_\lambda\left(\bigcup_{i=1}^{n} (A_i \cap A_{n+1})\right)}\left(Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) + Q_\lambda(A_{n+1}) + \right. \\ &\left. +\lambda Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) Q_\lambda(A_{n+1}) - Q_\lambda\left(\bigcup_{i=1}^{n} (A_i \cap A_{n+1})\right)\right). \end{aligned} \tag{13}$$

Now, by introducing $A_i^* = A_i \cap A_{n+1}$ for all $1 \le i \le n$, Eq. (13) can be written as

$$\begin{aligned} Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right) = \frac{1}{1+\lambda Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right)}&\left(Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) + Q_\lambda(A_{n+1}) + \right. \\ &\left. +\lambda Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) Q_\lambda(A_{n+1}) - Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right)\right). \end{aligned} \tag{14}$$

Next, using the inductive condition and the fact that $Z_{n,\lambda}^{(k)} = p_{n,\lambda}^{(k)}(A_1, \ldots, A_n)$ holds by definition for any $1 \le k \le n$, $Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right)$ can be written as

$$Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right) = \frac{1}{\lambda}\left(\prod_{k=1}^{n}\left(p_{n,\lambda}^{(k)}\left(A_1^*, \ldots, A_n^*\right)\right)^{(-1)^{k-1}} - 1\right).$$

Since $Z_{n,\lambda}^{*(k)} = p_{n,\lambda}^{(k)}(A_1^*, \ldots, A_n^*)$ holds by definition for any $1 \le k \le n$, the previous equation can be rewritten in the following form:

$$Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right) = \frac{1}{\lambda}\left(\prod_{k=1}^{n}\left(Z_{n,\lambda}^{*(k)}\right)^{(-1)^{k-1}} - 1\right). \tag{15}$$

Recall that based on Lemma 3, we have the following equation

$$Z_{n,\lambda}^{*(k)} = \begin{cases} \dfrac{Z_{n+1,\lambda}^{(k+1)}}{Z_{n,\lambda}^{(k+1)}}, & \text{if } k < n \\ Z_{n+1,\lambda}^{(n+1)}, & \text{if } k = n. \end{cases} \tag{16}$$

Applying Eq. (16) to Eq. (15) yields

$$\begin{gathered} Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right) = \\ = \frac{1}{\lambda}\left(\left(\prod_{k=1}^{n-1}\left(Z_{n+1,\lambda}^{(k+1)}\right)^{(-1)^{k-1}}\left(Z_{n,\lambda}^{(k+1)}\right)^{(-1)^{k}}\right)\left(Z_{n+1,\lambda}^{(n+1)}\right)^{(-1)^{n+1}} - 1\right). \end{gathered} \tag{17}$$

Next, based on Lemma 4,

$$Q_\lambda(A_{n+1}) = \frac{1}{\lambda}\left(\frac{Z_{n+1,\lambda}^{(1)}}{Z_{n,\lambda}^{(1)}} - 1\right). \tag{18}$$

Now, applying the inductive condition given in Eq. (11) and substituting the formulas for $Q_\lambda\left(\bigcup_{i=1}^{n} A_i^*\right)$ and $Q_\lambda(A_{n+1})$ given by Eq. (17) and Eq. (18), respectively, into Eq. (14) gives us

$$\begin{gathered} Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right) = \\ = \frac{\frac{1}{\lambda}(Y_1 - 1) + \frac{1}{\lambda}(Y_2 - 1) + \lambda\frac{1}{\lambda}(Y_1 - 1)\frac{1}{\lambda}(Y_2 - 1) - \frac{1}{\lambda}(Y_3 - 1)}{Y_3}, \end{gathered} \tag{19}$$

where

$$Y_1 = \prod_{k=1}^{n}\left(Z_{n,\lambda}^{(k)}\right)^{(-1)^{k-1}}$$

$$Y_2 = \frac{Z_{n+1,\lambda}^{(1)}}{Z_{n,\lambda}^{(1)}}$$

$$Y_3 = \left(\prod_{k=1}^{n-1}\left(Z_{n+1,\lambda}^{(k+1)}\right)^{(-1)^{k-1}}\left(Z_{n,\lambda}^{(k+1)}\right)^{(-1)^{k}}\right)\left(Z_{n+1,\lambda}^{(n+1)}\right)^{(-1)^{n+1}}.$$

Simplifying Eq. (19) leads to

$$Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right) = \frac{1}{\lambda}\left(\frac{Y_1 Y_2}{Y_3} - 1\right). \tag{20}$$

Now, by substituting the definitions of Y_1, Y_2 and Y_3 into Eq. (20), after simplifi-

cation we get

$$Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right) =$$

$$= \frac{1}{\lambda}\left(\frac{\left(\prod_{k=1}^{n}\left(Z_{n,\lambda}^{(k)}\right)^{(-1)^{k-1}}\right)\frac{Z_{n+1,\lambda}^{(1)}}{Z_{n,\lambda}^{(1)}}}{\left(\prod_{k=1}^{n-1}\left(Z_{n+1,\lambda}^{(k+1)}\right)^{(-1)^{k-1}}\right)\left(\prod_{k=1}^{n-1}\left(Z_{n,\lambda}^{(k+1)}\right)^{(-1)^{k}}\right)\left(Z_{n+1,\lambda}^{(n+1)}\right)^{(-1)^{n+1}}} - 1\right) =$$

$$= \frac{1}{\lambda}\left(\frac{Z_{n+1,\lambda}^{(1)}}{\left(\prod_{k=1}^{n-1}\left(Z_{n+1,\lambda}^{(k+1)}\right)^{(-1)^{k-1}}\right)\left(Z_{n+1,\lambda}^{(n+1)}\right)^{(-1)^{n+1}}} - 1\right) =$$

$$= \frac{1}{\lambda}\left(\frac{Z_{n+1,\lambda}^{(1)}}{\left(Z_{n+1,\lambda}^{(2)}\right)\left(Z_{n+1,\lambda}^{(3)}\right)^{-1}\cdots\left(Z_{n+1,\lambda}^{(n+1)}\right)^{(-1)^{n+1}}} - 1\right) =$$

$$= \frac{1}{\lambda}\left(\left(Z_{n+1,\lambda}^{(1)}\right)\left(Z_{n+1,\lambda}^{(2)}\right)^{-1}\left(Z_{n+1,\lambda}^{(3)}\right)\cdots\left(Z_{n+1,\lambda}^{(n+1)}\right)^{(-1)^{n}} - 1\right) =$$

$$= \frac{1}{\lambda}\left(\prod_{k=1}^{n+1}\left(Z_{n+1,\lambda}^{(k)}\right)^{(-1)^{k-1}} - 1\right).$$

Notice that this formula is the same as the formula for $Q_\lambda\left(\bigcup_{i=1}^{n+1} A_i\right)$ given in Eq. (12), which means that we have proved this theorem. □

On the one hand, Theorem 1 tells us how to compute the λ-additive measure of union of n sets in the case when λ is nonzero. On the other hand, it immediately follows from the definition of λ-additive measure that if $\lambda = 0$, then the λ-additive measure on the finite set X is a probability measure on X. Hence, if X is a finite set, $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \geq 2$, Q_λ is a λ-additive measure on X and $\lambda = 0$, then $Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right)$ can be computed by using the Poincaré formula of probability theory:

$$Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{k=1}^{n}(-1)^{k-1}\sum_{1\leq i_1<\cdots<i_k\leq n} Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right). \tag{21}$$

The following theorem shows how the Poincaré formula of probability theory given in Eq. (21) may be viewed as a limit case of the general formula of λ-additive measure of the union of n sets given in Eq. (10).

Theorem 2. *If X is a finite set, $A_1, \ldots, A_n \in \mathcal{P}(X)$, $n \geq 2$, Q_λ is a λ-additive measure on X and $\lambda \neq 0$, then*

$$\lim_{\lambda\to 0} Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{k=1}^{n}(-1)^{k-1}\sum_{1\leq i_1<\cdots<i_k\leq n} Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right). \tag{22}$$

Proof. Let $\lambda \neq 0$. Here, we will distinguish two cases. Namely, (1) when n is even; and (2) when n is odd.

(1) If n is even, then based on Theorem 1,

$$\lim_{\lambda\to 0} Q_\lambda\left(\bigcup_{i=1}^{n} A_i\right) = \lim_{\lambda\to 0}\left(\frac{1}{\lambda}\left(\frac{Z_{n,\lambda}^{(1)} Z_{n,\lambda}^{(3)} \cdots Z_{n,\lambda}^{(n-1)}}{Z_{n,\lambda}^{(2)} Z_{n,\lambda}^{(4)} \cdots Z_{n,\lambda}^{(n)}} - 1\right)\right) = \\ = \frac{\lim_{\lambda\to 0}\left(\frac{1}{\lambda}\left(Z_{n,\lambda}^{(1)} Z_{n,\lambda}^{(3)} \cdots Z_{n,\lambda}^{(n-1)} - Z_{n,\lambda}^{(2)} Z_{n,\lambda}^{(4)} \cdots Z_{n,\lambda}^{(n)}\right)\right)}{\lim_{\lambda\to 0}\left(Z_{n,\lambda}^{(2)} Z_{n,\lambda}^{(4)} \cdots Z_{n,\lambda}^{(n)}\right)}, \tag{23}$$

where

$$Z_{n,\lambda}^{(k)} = \prod_{1\leq i_1<\cdots<i_k\leq n} \left(1 + \lambda Q_\lambda\left(A_{i_1} \cap \cdots \cap A_{i_k}\right)\right),$$

$1 \leq k \leq n$. Definition of $Z_{n,\lambda}^{(k)}$ implies that

$$\lim_{\lambda\to 0}\left(Z_{n,\lambda}^{(2)} Z_{n,\lambda}^{(4)} \cdots Z_{n,\lambda}^{(n)}\right) = 1. \tag{24}$$

Let $F(\lambda; A_1, \ldots, A_n) = Z_{n,\lambda}^{(1)} Z_{n,\lambda}^{(3)} \cdots Z_{n,\lambda}^{(n-1)} - Z_{n,\lambda}^{(2)} Z_{n,\lambda}^{(4)} \cdots Z_{n,\lambda}^{(n)}$. Applying the definition of $Z_{n,\lambda}^{(k)}$, after direct calculations we get

$$F(\lambda; A_1, \ldots, A_n) = \\ 1 + \sum_{1\leq i\leq n} \lambda Q_\lambda(A_i) + \sum_{1\leq i_1<i_2<i_3\leq n} \lambda Q_\lambda(A_{i_1} \cap A_{i_2} \cap A_{i_3}) + \cdots \\ \cdots + \sum_{1\leq i_1<\cdots<i_{n-1}\leq n} \lambda Q_\lambda(A_{i_1} \cap \cdots \cap A_{i_{n-1}}) + G(\lambda) - \\ -1 - \sum_{1\leq i_1<i_2\leq n} \lambda Q_\lambda(A_{i_1} \cap A_{i_2}) - \sum_{1\leq i_1<i_2<i_3<i_4\leq n} \lambda Q_\lambda(A_{i_1} \cap A_{i_2} \cap A_{i_3} \cap A_{i_4}) - \cdots \\ \cdots - \sum_{1\leq i_1<\cdots<i_n\leq n} \lambda Q_\lambda(A_{i_1} \cap \cdots \cap A_{i_n}) - H(\lambda),$$

where $G(\lambda)$ and $H(\lambda)$ are at least second order polynomials of λ in which the constant term is equal to zero. Thus,

$$\lim_{\lambda\to 0}\left(\frac{1}{\lambda} G(\lambda)\right) = 0, \qquad \lim_{\lambda\to 0}\left(\frac{1}{\lambda} H(\lambda)\right) = 0$$

and so

$$\lim_{\lambda\to 0}\left(\frac{1}{\lambda}F(\lambda;A_1,\ldots,A_n)\right)=$$
$$=\sum_{1\le i\le n}Q_\lambda(A_i)-\sum_{1\le i_1<i_2\le n}Q_\lambda(A_{i_1}\cap A_{i_2})+$$
$$+\sum_{1\le i_1<i_2<i_3\le n}Q_\lambda(A_{i_1}\cap A_{i_2}\cap A_{i_3})-\sum_{1\le i_1<i_2<i_3<i_4\le n}Q_\lambda(A_{i_1}\cap A_{i_2}\cap A_{i_3}\cap A_{i_4})+$$
$$\cdots$$
$$+\sum_{1\le i_1<\cdots<i_{n-1}\le n}Q_\lambda(A_{i_1}\cap\cdots\cap A_{i_{n-1}})-\sum_{1\le i_1<\cdots<i_n\le n}Q_\lambda(A_{i_1}\cap\cdots\cap A_{i_n}).$$

That is, we have the following equation:

$$\begin{aligned}\lim_{\lambda\to 0}\left(\frac{1}{\lambda}F(\lambda;A_1,\ldots,A_n)\right)&=\\ =\lim_{\lambda\to 0}\left(\frac{1}{\lambda}\left(Z_{n,\lambda}^{(1)}Z_{n,\lambda}^{(3)}\cdots Z_{n,\lambda}^{(n-1)}-Z_{n,\lambda}^{(2)}Z_{n,\lambda}^{(4)}\cdots Z_{n,\lambda}^{(n)}\right)\right)&=\\ =\sum_{k=1}^{n}(-1)^{k-1}\sum_{1\le i_1<\cdots<i_k\le n}Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right).&\end{aligned}\tag{25}$$

Now, by substituting the formulas in Eq. (24) and Eq. (25) into Eq. (23), we get

$$\lim_{\lambda\to 0}Q_\lambda\left(\bigcup_{i=1}^{n}A_i\right)=\sum_{k=1}^{n}(-1)^{k-1}\sum_{1\le i_1<\cdots<i_k\le n}Q_\lambda\left(A_{i_1}\cap\cdots\cap A_{i_k}\right).$$

(2) In the case where n is an odd number, the theorem can be proved by following steps similar to those of case (1). □

This result tells us that our general formula for the λ-additive measure of the union of n sets may be viewed as the generalization of the Poincaré formula of probability theory.

3 Summary and future plans

The key findings of this study can be summarized as follows.

(1) We presented the general formula for the λ-additive measure of the union of n sets in Eq.(1), and gave an elementary proof of it in Theorem 1.

(2) Using elementary techniques, we demonstrated that the Poincaré formula of probability theory given in Eq. (2) is just a limit case of the general formula for the λ-additive measure of the union of n sets given in Eq. (1); that is, our formula may be viewed as a generalization of the Poincaré formula.

In the future, we should like to formulate a calculus of the λ-additive measure and generalize the Bayes theorem for λ-additive measures. We also plan to study how the λ-additive measure and the generalized Poincaré formula can be utilized in the fields of computer science, engineering and economics.

References

[1] Chen, Xing, Huang, Yu-An, Wang, Xue-Song, You, Zhu-Hong, and Chan, Keith CC. FMLNCSIM: fuzzy measure-based lncRNA functional similarity calculation model. *Oncotarget*, 7(29):45948–45958, 2016. DOI: 10.18632/oncotarget.10008.

[2] Chiţescu, Ion. Why λ-additive (fuzzy) measures? *Kybernetika*, 51(2):246–254, 2015. DOI: 10.14736/kyb-2015-2-0246.

[3] Dempster, A. P. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967. DOI: 10.1214/aoms/1177698950.

[4] Dombi, József and Jónás, Tamás. The general Poincaré formula for λ-additive measures. *Information Sciences*, 490:285–291, 2019. DOI: 10.1016/j.ins.2019.03.059.

[5] Dubois, Didier and Prade, Henri. *Fuzzy Sets and Systems: Theory and Applications*, volume 144 of *Mathematics In Science And Engineering*, chapter 5, pages 125–147. Academic Press, Inc., Orlando, FL, USA, 1980.

[6] Dubois, Didier and Prade, Henri. Rough fuzzy sets and fuzzy rough sets. *International Journal of General Systems*, 17(2–3):191–209, 1990. DOI: 10.1080/03081079008935107.

[7] Feng, Tao, Mi, Ju-Sheng, and Zhang, Shao-Pu. Belief functions on general intuitionistic fuzzy information systems. *Information Sciences*, 271:143–158, 2014. DOI: 10.1016/j.ins.2014.02.120.

[8] Grabisch, Michel. *Set Functions, Games and Capacities in Decision Making*. Springer Publishing Company, Incorporated, 1st edition, 2016. DOI: 10.1007/978-3-319-30690-2_2.

[9] Höhle, Ulrich. A general theory of fuzzy plausibility measures. *Journal of Mathematical Analysis and Applications*, 127(2):346–364, 1987. DOI: 10.1016/0022-247X(87)90114-4.

[10] Jin, LeSheng, Mesiar, Radko, and Yager, Ronald R. Melting probability measure with owa operator to generate fuzzy measure: the crescent method. *IEEE Transactions on Fuzzy Systems*, 27(6):1309–1316, 2018. DOI: 10.1109/tfuzz.2018.2877605.

[11] Magadum, C.G. and Bapat, M.S. Ranking of students for admission process by using Choquet integral. *International Journal of Fuzzy Mathematical Archive*, 15(2):105–113, 2018.

[12] Mohamed, M. A. and Xiao, Weimin. Q-measures: an efficient extension of the Sugeno λ-measure. *IEEE Transactions on Fuzzy Systems*, 11(3):419–426, 2003. DOI: 10.1109/tfuzz.2003.812701.

[13] Pap, Endre. *Null-additive set functions*, volume 337. Kluwer Academic Pub, 1995.

[14] Pap, Endre. Pseudo-additive measures and their applications. In *Handbook of measure theory*, pages 1403–1468. Elsevier, 2002. DOI: 10.1016/b978-044450263-6/50036-1.

[15] Polkowski, Lech. *Rough sets in knowledge discovery 2: applications, case studies and software systems*, volume 19. Physica, 2013.

[16] Shafer, Glenn. *A mathematical theory of evidence*, volume 42. Princeton University Press, 1976.

[17] Singh, Akhilesh Kumar. Signed λ-measures on effect algebras. In *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, pages 1–7. Springer India, Jul 2018. DOI: 10.1007/s40010-018-0510-x.

[18] Skowron, Andrzej. The relationship between the rough set theory and evidence theory. *Bulletin of Polish academy of science: Mathematics*, 37:87–90, 1989.

[19] Skowron, Andrzej. The rough sets theory and evidence theory. *Fundam. Inf.*, 13(3):245–262, October 1990.

[20] Spohn, Wolfgang. *The Laws of Belief: Ranking Theory and its Philosophical Applications*. Oxford University Press, 2012. DOI: 10.1093/acprof:oso/9780199697502.001.0001.

[21] Sugeno, M. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, Tokyo, Japan, 1974.

[22] Wang, Zhenyuan and Klir, George J. *Generalized Measure Theory*. IFSR International Series in Systems Science and Systems Engineering. Springer US, 2010.

[23] Wu, Wei-Zhi, Leung, Yee, and Zhang, Wen-Xiu. Connections between rough set theory and Dempster-Shafer theory of evidence. *International Journal of General Systems*, 31(4):405–430, 2002. DOI: 10.1080/0308107021000013626.

[24] Yao, Y.Y. and Lingras, P.J. Interpretations of belief functions in the theory of rough sets. *Information Sciences*, 104(1):81–106, 1998. DOI: 10.1016/S0020-0255(97)00076-5.

Received 28th March 2019

Acta Cybernetica **24** (2019) 187–195.

Linear Time Ordering of Bins using a Conveyor System*

Géza Makay[a] and András Pluhár[b]

Abstract

A local food wholesaler company is using an automated commissioning system, which brings the bins containing the appropriate product to the commissioning counter, where the worker picks the needed amounts to 12 bins corresponding to the same number of orders. To minimize the number of bins to pick from, they pick for several different spreading tours, so the order of bins containing the picked products coming from the commissioning counter can be considered random in this sense. Recently, the number of bins containing the picked orders increased over the available storage space, and it was necessary to find a new way of storing and ordering the bins to spreading tours. We developed a conveyor system which (after a preprocessing step) can order the bins in linear space and time.

Keywords: material flow control, bin ordering, modified Yehuda-Fogel algorithm

1 Introduction

Automated material handling systems (AMHSs) are used in several different areas throughout the world: baggage handling, distribution, postal services, etc. The different market sectors have different goals and challenges, so it is very hard to create a common platform for all of them. There are several approaches to cope with this problem, see for example Haneyah et al. [4] and the references therein.

This paper is motivated by a collaboration of the authors and a local food wholesaler company. The company is using an automated commissioning system, where 3 PLC-controlled robots (each serving from 2 rows on their left and right) bring out and take the bins into 6048 storage spaces. The bins travel through a PLC-controlled conveyor system to the commissioning counters, where the workers

*This research was supported by the Ministry of Human Capacities, Hungary [grant 20391-3/2018/FEKUSTRAT] and by the National Research, Development and Innovation Office - NKFIH [grant SNN-117879].

[a]Bolyai Institute, University of Szeged, Hungary, E-mail: makayg@math.u-szeged.hu

[b]Department of Computational Optimization, University of Szeged, Hungary, E-mail: pluhar@inf.u-szeged.hu

DOI: 10.14232/actacyb.24.2.2019.2

pick one product for several orders at the same time, which is faster than picking the product for one order only. This method decreases the number of needed bins to pick from, therefore decreases the work of the workers, the robots and the conveyor system. There is a controlling system over the PLC, called MFC (Material Flow Control) system, which (among others) optimizes the order of bins arriving to the commissioning places. When a bin is full or the order's picking is complete, the worker places the commissioned bin to the conveyor system, and the MFC brings it to one of the storage places. This also means, that the commissioned bins are coming out of the commissioning place in more-or-less random order, but it is not a problem: when a spreading tour is complete, the robots bring them out in the correct order. A spreading tour is practically an order of customers visited by one truck or van, the customers are chosen so that the truck's load is utilized as much as possible. And the correct order of the bins for one spreading tour is the reverse order of the customers' visiting order on the tour, since when the bins are placed on a pallet in this order and put in a truck or van, the truck or van works as a LIFO (last in first out) stack.

Recently, the number of commissioned bins increased over the available storage space, and the company needed a new way of storing and ordering the bins to spreading tours without using the robots. The problem is twofold: we need an affordable storage and sorting hardware, but a hardware which is capable of ordering up to 100 bins (that is the maximum number of bins on one spreading tour). The conveyor system we use for commissioning is quite expensive, since each module has its own electric motor, rolling cylinders to pass the bins on and some of the modules (where the bins need to change traveling directions) even have belts and a pneumatic system to lift the bins. On the other hand a conveyor belt is relatively cheap: one motor and a belt for 40-50 bins. So we decided to use conveyor belts wherever possible for storage, and a mix of conveyor belts and modules for ordering.

In Section 2 we describe the mathematical model for this problem. One of the best ordering methods using conveyor belts and modules is the merge sort, so our main problem is to create monotone subsequences from the original sequence of bins to use this sorting method on this hardware. In Section 3 we overview the relevant literature and give an example for such partitioning using the Erdős-Szekeres Theorem [3]. The modified Yehuda-Fogel algorithm [5] is able to find this partioning faster, but since the moving of bins is much slower than this preprocessing step, the total time of ordering mainly depends on the former. In Section 4 we show a possible realization of the physical system, which is able to reorder the bins in linear time and space.

2 Modeling the problem

The problem has quite a few connections to several areas in the literature. One of the most closely related topic is the parallel stack loading problem [1], which uses LIFO stack structures to store items in (preferably) decreasing order so that no blockages occur while unloading the stacks. However, generally blockages may

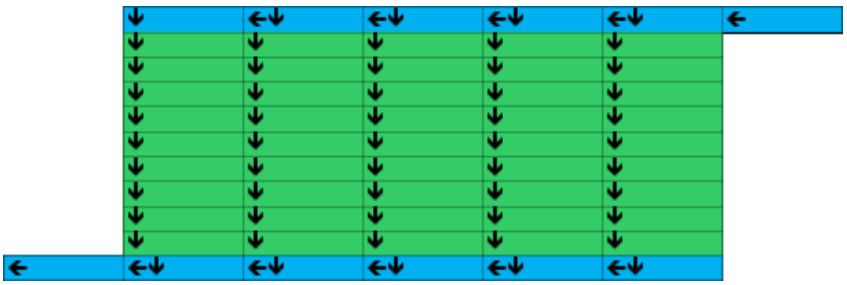


Figure 1: A storage setup

occur, in other words, the items cannot be stored such a way that they can be retrieved from the stacks in the desired order. In our case this is not allowed, we do need the correct order of bins. The main idea of the solution is to use parallel loading of LIFO and FIFO structures corresponding decreasing and increasing parts of the bin sequence to achieve this goal.

There are several methods to sort n numbers quickly, an obvious one is quicksort which runs in an expected $n \log n$ time. However, this method exchanges elements far from each other, it would need a complicated hardware and it would take a long time to perform the exchange physically, therefore it is not suitable for bin ordering. On the other hand merge-sort is just perfect for sorting bins on a conveyor system. To see this first we describe a relatively cheap conveyor system capable of storing and transporting bins and performing merge-sort.

A conveyor system is made up from two major parts: a conveyor belt moving the bins in one direction, and a *direction changer module*. A direction changer module can pick up a bin from one direction using (for example) a belt, and by lowering the belt deposits the bin to the rolling cylinders, which move the bin forward in another direction. A simple storage system is shown on Figure 1. The green modules are conveyor belts for storage, while the blue modules are the direction changer modules capable of forwarding the bins from the blue modules to the green ones and from the green ones to the blue ones. The bins are coming in from the upper-right direction and leaving the storage system through the lower-left module. The bins can be sorted to the green modules by spreading tours, but within one spreading tour their order is still not specified, can be considered random.

Let us define the ordering problem. We assume that the bins are on a conveyor belt in random order, and we want them on a conveyor belt in the correct order. One step of a conveyor belt is when it moves all bins on it one step further. One step of a direction changing module is when the module drops the bin to the next belt/module and picks up the next bin from the previous belt/module. Physically a direction changer module cannot perform these actions in parallel, but it would make computation much more complicated if we took this into account. However it does not change the order of steps needed for reordering the bins: it adds a factor of 2 in the very worst case.

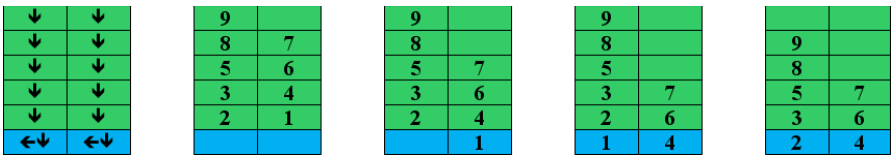


Figure 2: Merge-sort for 2 lines

3 The mathematical background of the problem

We proposed merge sorting for the physical hardware consisting of conveyor belts and direction changing modules. Our next theorem shows that an appropriate conveyor system is capable of performing merge-sort in linear time. Later we show that we can construct the monotone subsequences needed for the algorithm presented in this theorem.

Theorem 1. *Suppose, that we have k lines of ordered set of bins represented here by increasing numbers, and we want to merge that k lines into one ordered set of bins. We claim that this can be executed in at most $n + k + 1$ steps, where n is the number of bins. One step here is a movement of all bins that need to be moved from one slot to the next one.*

Proof. We prove our claim by induction. For $k = 1$ the statement is trivial, as the bins are already in one ordered set.

An example of the case $k = 2$ is shown on Figure 2. The first column shows the direction in which the conveyor system can move the bins. On the second part the original setup of the two ordered list of numbers, and the rest shows the first couple of merging steps. Once a bin is at the lower-left module, it is the next in the merged order, it is considered sorted, and it steps out of the merging system to the left. In each step that number goes to the lower-left position, which is smaller, and its line moves forward. It is easy to see that the first number appears in the lower-left position in one or two steps, and after that a number comes in each step. So the total number of steps to clear the sorting system of all bins is either $n + 2$ or $n + 3$ which gives the result for $k = 2$.

Suppose that we know the claim of the theorem for k lines. Let us consider the case of $k + 1$ lines. From our induction assumption we know that while merge-sorting the right k lines, the smallest number will appear in at most $k + 1$ steps in the lower-left position of that k lines. Then the merging continues as we have already shown for two lines, i.e. it takes at most one more step for the smallest of the $k + 1$ lines to appear at the lower-left position of the whole table, then the numbers are coming continuously, so all together the number of steps is at most $n + k + 2$. □

To use merge-sorting one needs already sorted numbers, i.e. if we have n numbers then we would need monotonic subsequences of that, as they can be stored on

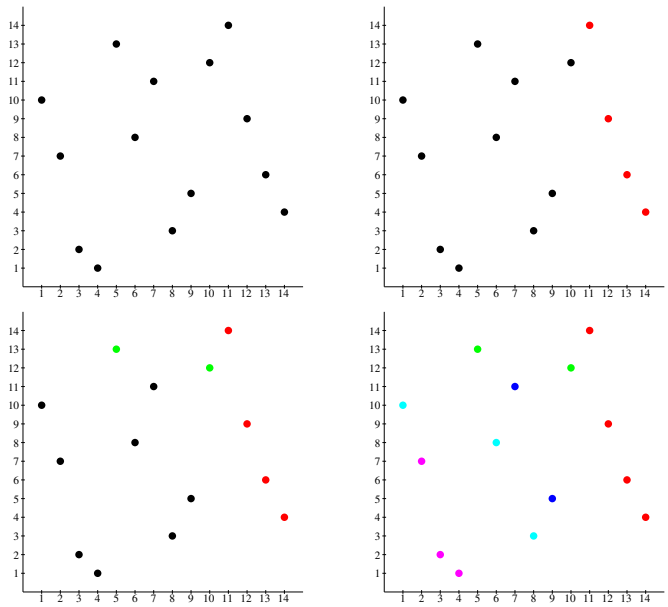


Figure 3: Illustration for finding long monotonic sequences

different lines and then merged. First we show a heuristics to find long monotone subsequences based on the proof of the Erdős and Szekeres Theorem [3].

Theorem 2. *For given r and s any sequence of distinct numbers with length at least $(r-1)(s-1)+1$ contains either a monotonically increasing subsequence of length r or a monotonically decreasing subsequence of length s.*

As a special case we get that a sequence of n numbers contains a monotonic subsequence of length $\lceil\sqrt{n}\,\rceil$ where $\lceil x\rceil$ is the smallest integer such that $x \leq \lceil x\rceil$.

Applying this result for the original sequence, then to the sequence from which the found monotonic subsequence is removed, etc, one can find monotonic subsequences of the original sequence so that all elements appear in at least one subsequence, i.e. it is a partition of the original sequence to monotonic subsequences. Let us see an algorithm to find a long enough monotonic subsequence. We demonstrate the algorithm on the sequence 10, 7, 2, 1, 13, 8, 11, 3, 5, 12, 14, 9, 6, 4. To illustrate this sequence, take the index of an element as a first coordinate, and the element itself as the second coordinate as shown in Figure 3a.

Let us find *peak elements* in the sequence: they are larger than any elements in the sequence after them. These elements correspond to points in the figure which do not have other points in their upper-right quarter. They are marked by red dots on Figure 3b and they are called the *Pareto border* or layer 1 of the point set. Peeling this Pareto border off, we find another set of peak elements (layer 2)

marked by green on Figure 3c. Continuing this procedure we color all points, all of them will be part of some layer. Now there are two possibilities according to the Theorem of Erdős and Szekeres.

- Either there is a long enough layer corresponding to a monotonically decreasing subsequence. Since we now have $n = 14$ elements,"long enough" means now $\lceil\sqrt{14}\,\rceil = 4$ elements. Layer 1 satisfies this property.
- Or (if all layers have less than 4 elements) then there must be at least 4 layers, which is also true now: we have 5 layers. In this case if we choose one point from layer 5 it is not a peak element considering the points of layer 5 and 4, so there must be a point in layer 4 which is in its upper-right quarter. Then this point is not peak in layer 5, 4 and 3, so there is a point in layer 3 in its upper-right quarter. Continuing this procedure we find one point from each layer forming a monotonically increasing subsequence of length 5.

By removing this long enough monotonic subsequence from the sequence and repeating this procedure one can arrive (for example) to this partitioning: 10, 7, 2, 1, 13, 8, 11, 3, 5, 12, 14, 9, 6, 4. Our algorithm takes $O(n^3)$ time to complete, and it shows the basic method for finding a partitioning of a sequence to monotonic subsequences. The partitioning guaranteed by the Erdős and Szekeres Theorem can be constructed by an asymptotically better algorithm developed by Yehuda and Fogel [6].

Theorem 3. *A sequence of n numbers can be partitioned into $2\lfloor\sqrt{n}\rfloor$ monotone subsequences in time $O(n^{1.5})$. All the subsequences can be chosen to be of size $\lceil\sqrt{n}\,\rceil$ or less.*

Brandstädt and Kratsch [2] gave the smaller bound of $\left\lfloor\sqrt{2n+1/4}-1/2\right\rfloor$ on the number of subsequences and proved that it is a tight bound. Recently Yang et al. [5] modified the Yehuda-Fogel algorithm to provide at most $\left\lfloor\sqrt{2n+1/4}-1/2\right\rfloor$ monotonic subsequences of size no more than $\lceil\sqrt{n}\,\rceil$ in $O(n^{1.5})$ time.

If our algorithm has to comply with the second part of the theorem above, then in our example we can simply ignore the extra elements in the monotonic subsequences we find during the procedure. Then we would get the following partitioning for example: 10, 7, 2, 1, 13, 8, 11, 3, 5, 12, 14, 9, 6, 4.

4 The physical realization of the sorting system

Now we have the theoretical background to design the physical sorting system.

We assume, that the bins are on a conveyor belt one after the other. Their desired order is represented by numbers. First we apply the modified Yehuda-Fogel algorithm to partition them into monotone subsequences. Since the number of bins are relatively small (in our case under 100), their algorithm runs within milliseconds on a modern computer. Then let us feed the bins from the lower-right module into the construction shown on Figure 4. This construction has

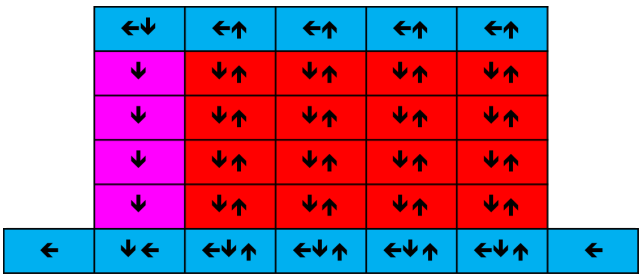


Figure 4: Construction of the sorting conveyor system

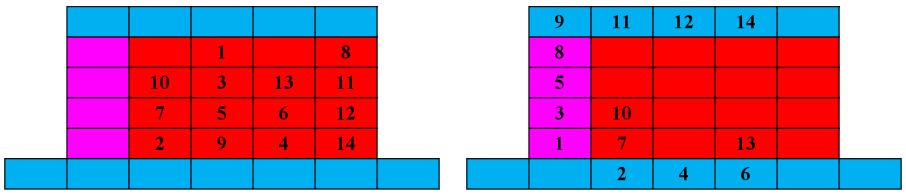


Figure 5: Using the sorting conveyor system

$\left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor$ columns of red sorting conveyor belts, each belt has the length $\lceil \sqrt{n} \rceil$ (the figure shows the numbers for our example sequence: for $n = 14$ we have $\left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor = 4$ columns of length $\lceil \sqrt{n} \rceil = 4$). Each monotonic subsequence goes into its own red conveyor belt: there are enough belts and they are long enough to accommodate them (see Figure 5a). This partitioning of the bins takes at most $n + \left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + \lceil \sqrt{n} \rceil$ bin moving steps. Monotonically increasing subsequences go up in their red conveyor belts, while monotonically decreasing sequences remain down. Then the system merge-sorts the monotonically increasing subsequences by moving them upward out of the red conveyor belts (see Theorem 1). At the same time the system merge-sorts the monotonically decreasing subsequences by moving them downward out of the red conveyor belts, which makes the merged sequence monotonically increasing (see Figure 5b). Note that on the figure the monotonically decreasing sequences are in the middle of sorting as they need to wait for the monotonically increasing sequence to arrive. Finally the system merge-sorts these two subsequences to produce the final sorted sequence. The last sortings take at most $n + \left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + \lceil \sqrt{n} \rceil + 1$ steps. So all together the sorting needs $2\left(n + \left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + \lceil \sqrt{n} \rceil\right) + 1$ time to complete, that is, it is linear in the number of bins.

The space required to order n bins is $\left(\left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + 1\right)(\lceil \sqrt{n} \rceil + 2)$

which is approximately $n\sqrt{2}$ so the space requirement is also linear in the number of bins.

In a physical setup one does not need the complete sorted sequence of bins standing on a conveyor belt, for example the bins can be moved to a pallet while they are still coming out of the sorting system. Therefore the actual time from the start of the sorting to the point when one can start working and (according to Theorem 1) can continuously work with the sorted sequence is at most $n + 2\left\lfloor\sqrt{2n+1/4} - 1/2\right\rfloor + 2\lceil\sqrt{n}\,\rceil + 1$.

This sorting conveyor system can sort more than n bins, if the sequence can be partitioned so that its monotonic subsequences fit into the sorting system. This can be achieved (for example) by pre-merge-sorting the bins while they are coming out of the storage conveyor belts. With this change one can achieve that all monotonic subsequences are decreasing, and hence we do not even need the upper (blue) and the left (purple) modules and belts containing $\left\lfloor\sqrt{2n+1/4} - 1/2\right\rfloor + 2$ expensive direction changing modules: a cut in the costs. Moreover, the lengths of the subsequences can be maximized, so the sorting system can be completely occupied during the sorting.

The sorting can run even faster by installing several (say: m) of these kind of sorting systems, dividing the original sequence into m subsequences of more-or-less equal size, sorting them parallel in the sorting systems and then merging the sorted subsequences into one sequence.

5 Conclusions

The problem of ordering bins seems to be neglected in the literature. There are several sorting agents, which can collect items to different containers based on some of their property, but they do not order those items within one container. We constructed a system which solves the ordering problem, so that the number of bin moving steps and the required space is linear in the number of bins. Searching for the modified Yehuda-Fogel algorithm (which is essential for this method to work) did not return an application similar to the one presented in this paper, so this method of ordering the bins looks novel. The system itself is a generalization of the parallel loading of LIFO stacks by adding FIFO capabilities to the stacks.

Although the physical ordering of n bins takes linear time in the number of bins, we should not forget that we need the preliminary task of partitioning the sequence into monotone subsequences, and it takes $O(n^{1.5})$ time. However, the physical ordering of bins is much slower: our conveyor system moves the bins by at most 1 module/second speed. If we had, say, 1 million bins (which is not realistic in practice), it would take days to order them even when using much faster conveyor belts and modules, so the time needed for the partitioning is negligible compared to this time frame.

References

[1] Boysen, Nils and Emde, Simon. The parallel stack loading problem to minimize blockages. *European Journal of Operational Research*, 249(2):618–627, 2016. DOI: 10.1016/j.ejor.2015.09.033.

[2] Brandstädt, Andreas and Kratsch, Dieter. On partitions of permutations into increasing and decreasing subsequences. *Elektronische Informationsverarbeitung und Kybernetik*, 22(5/6):263–273, 1986.

[3] Erdős, Pál and Szekeres, György. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[4] Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., and Zijm, W.H.M. Generic planning and control of automated material handling systems. *Computers in Industry*, 64(3):177–190, 2013. DOI: 10.1016/j.compind.2012.11.003.

[5] Yang, Bing, Chen, Jing, Lu, En-Yue, and Zheng, Si-Qing. Design and performance evaluation of sequence partition algorithms. *Journal of Computer Science and Technology*, 23(5):711–718, 2008. DOI: 10.1007/s11390-008-9183-2.

[6] Yehuda, Reuven Bar and Fogel, Sergio. Partitioning a sequence into few monotone subsequences. *Acta Informatica*, 35(5):421–440, 1998. DOI: 10.1007/s002360050126.

Received 22nd March 2019

Acta Cybernetica **24** (2019) 197–209.

Automating, Analyzing and Improving Pupillometry with Machine Learning Algorithms

György Kalmár[a], Alexandra Büki[b], Gabriella Kékesi[b], Gyöngyi Horváth[b], and László G. Nyúl[a]

Abstract

The investigation of the pupillary light reflex (PLR) is a well-known method to provide information about the functionality of the autonomic nervous system. Pupillometry, a non-invasive technique, was applied to study the PLR alterations in a new, schizophrenia-like rat substrain, named WISKET. The pupil responses to light impulses were recorded with an infrared camera; the videos were automatically processed and features were extracted from the pupillograms. Besides the classical statistical analysis (ANOVA), feature selection and classification were applied to reveal the significant differences in the PLR parameters between the control and WISKET animals. Based on these results, the disadvantages of this method were analyzed and the measurement setup was redesigned and improved. The pupil segmentation method has also been adapted to the new videos. 2564 images were annotated manually and used to train a fully-convolutional neural network to produce pupil mask images. The method was evaluated on 329 test images and achieved 4% median relative error. With the new setup, the pupil detection became reliable and the new data acquisition offers robustness to the experiments.

Keywords: pupillometry, classification, curve properties, U-Net

1 Introduction

Patients with schizophrenia, besides the well-known behavioral symptoms, also show autonomic dysregulation, including impaired pupillary function, which is a sensitive and reliable source of information about the function of the nervous system [1].

Pupillometry is a simple, non-invasive technique for the assessment of the autonomic nervous system function by testing the pupillary light reflex (PLR), meaning

[a]Department of Image Processing and Computer Graphics, Faculty of Science and Informatics, University of Szeged, E-mail: {kalmargy,nyul}@inf.u-szeged.hu

[b]Department of Physiology, Faculty of Medicine, University of Szeged, E-mail: {buki.alexandra,kekesi.gabriella,horvath.gyongyi}@med.u-szeged.hu

DOI: 10.14232/actacyb.24.2.2019.3

the contraction of the pupil in response to light. During the test, the changes of the pupil diameter are recorded and the size of the pupil is measured offline in each video frame, producing the *pupillogram*.

The measurement of the diameter in each frame manually is a labor-intensive and slow process. Automated, software-based methods can speed-up the detection and improve the analyzing process.

Our recent research investigated the PLR to reveal the schizophrenia-related alterations in the autonomic nervous system of WISKET rats and the related *medical* results were published in [3]. Contributions in the current work: the explanation of novel pupillogram features; presentation of a decision tree based classifier and the discussion of the results; introduction of a redesigned measurement setup and process; description and evaluation of the new pupil segmentation method.

Related works are summarized in Section 2. The experiments, the novelties of the feature extraction method and data analysis are described in Section 3. In Section 4 the results of the analysis and classification are presented and discussed. Section 5 describes the redesigned measurement setup and a manually annotated pupil segmentation dataset. A deep learning based pupil detection algorithm and the detection results on a test dataset are shown in Section 6. Section 7 concludes the paper.

2 Related works

Developing reliable and predictive animal models for any complex psychiatric diseases, such as schizophrenia, is essential to understand the neurobiological basis of the disorder. Recently, a new, selectively bred rat model of schizophrenia has been developed, named WISKET [7, 8, 11, 19].

Clinical studies in schizophrenic patients using pupillometry revealed impaired autonomic regulation [1, 2, 6, 13, 22]. PLR was also investigated in rodents [16]; however, only our recent study provided data from animal models of schizophrenia [3].

Speeding-up pupillometry with automated softwares is essential because manual methods are slow and labor-intensive. In cases, when the image quality is acceptable, circle or ellipse detection algorithms can be used [15, 21]. These methods rely on simple techniques, e.g. the Hough-transform with an ellipse model. When low contrast and reflections occur, complex and more sophisticated algorithms are needed [14, 23]. These solutions combine several simple methods with specialized extensions, e.g. in [14], the simple Hough-transform based ellipse detection was extended with randomization and was implemented in an iterative scheme to filter out the noise and to handle outliers. The recent rapid development of artificial intelligence and machine learning led to more accurate solutions to the pupil detection problem [4, 5]. Among these methods, the convolutional neural networks are particularly interesting as they can treat their inputs as images. When the expected outputs are binary masks, which is a common goal in biomedical applications, fully-convolutional neural networks are used. They utilize 'de-convolutional' layers to

up-sample and combine feature maps with low resolutions to binary masks having the same sizes as the input images. The most popular such a network structure, the U-Net, was introduced in [20].

3 Automated pupillometry

3.1 Data acquisition

As it was described recently [3], two series of experiments were performed in sedated (n=54) and anesthetized (n=20) control Wistar, and WISKET rats. After a 10-minute long dark adaptation period, the recordings lasted for $15s$ in sedated, and for $60s$ in anesthetized animals. The animals were positioned close to a camera, and an intensive visible light stimulus (approx. $300cd/m^2$ for $600ms$) was flashed into their left eyes. The IR-camera recorded pupillary responses at a speed of 24 frames-per-second under infrared illumination.

During the PLR measurements, the rats showed several minor movements, which affected the quality of the recorded videos. Furthermore, albino rats lack pigments in their body including their eyes, which reduces the contrast between the iris and the pupil. A specifically designed pupil detection and measurement algorithm was used to handle these quality drawbacks [10]. The input of the algorithm was the video recording; and the output was a curve of the determined relative pupil diameters in each frame - the pupillogram. The relative pupil diameter is the ratio of the pupil and iris diameters, expressed in percentage.

3.2 Feature extraction

To compare the responses of the animals, descriptive features from the PLR curves were extracted, which are relevant from the pathophysiological point of view and suitable to emphasize the differences between the groups regarding the autonomic nervous system activities.

An automated feature extraction method was designed, which produced 40 features from the pupillogram. Many of them were basic, traditional parameters like the initial diameter, which is the size of the pupil before the light impulse; minimum diameter; reaction time; maximum of the redilated diameter; etc. Several new features were implemented to obtain information about the dynamics of the response, thus 11 velocity related descriptors were introduced including, the average and the maximal contraction velocities and the times required to reach the latter. The velocities of the redilation phase at different time points were also calculated.

Novel, smoothness related descriptors were introduced, as well. A polynomial curve with a given order was fitted to the redilation part of the response. The area between the original and fitted curves served as a measure of non-smoothness. Fifth order polynomials were chosen, which were flexible enough to follow the slow perturbations of the original curve and indicated only the short, abnormal swings. In Figure 1, a representative response curve and a marked subset of the extracted features are presented.

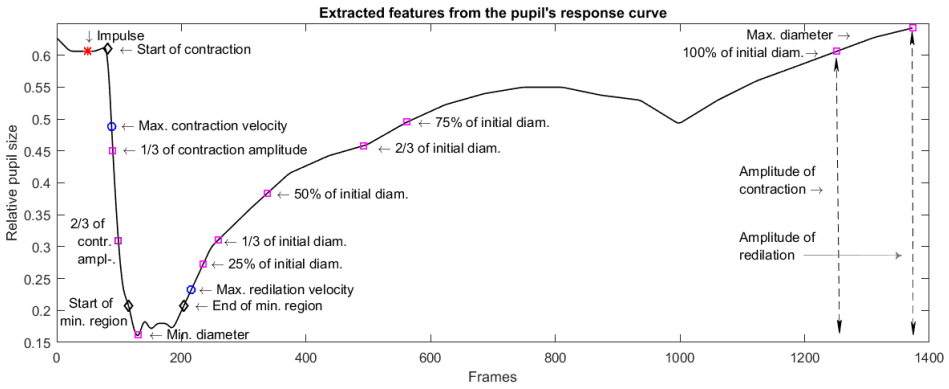


Figure 1: Pupillary light response curve and a marked subset of the 40 features.

3.3 Data analysis

The sedated and the anesthetized animals were analyzed separately.

One-way ANOVA was used for the analysis of differences between the control and test groups. The relationships between pupil parameters were assessed by linear regression analysis and calculation of the Pearson correlation coefficient. Only probabilities lower than 0.05 were considered significant.

Besides the investigation of the differences in PLR parameters, the other goal was to use pupillometry as a quick examination to facilitate the selection of the animals during the breeding process. Therefore, a binary decision tree was trained to investigate the possibility of classification and the model was evaluated by using cross-validation.

4 Evaluation

4.1 Classification results

The detailed discussion of the results of the statistical analysis is found in our recent study [3]. In accordance with these results, the trained decision tree selected almost the same features as predictor variables as the statistical analysis suggested.

In the sedated group, which has the greater cardinality, the fitted decision tree achieved 71% accuracy measured by cross-validation. The algorithm selected the following predictors: minimum diameter (relative pupil size); initial diameter (relative pupil size), and average redilation speed (change of relative pupil size/frame time). Figure 2 shows the fitted decision tree.

With the combination of the two analysis methods (statistical analysis, decision tree), significant differences were noticed between the control and test groups. The initial and minimum pupil diameters were larger and the degree of the constriction was lower in the WISKET rats. The flatness of the curve (length of the minimum

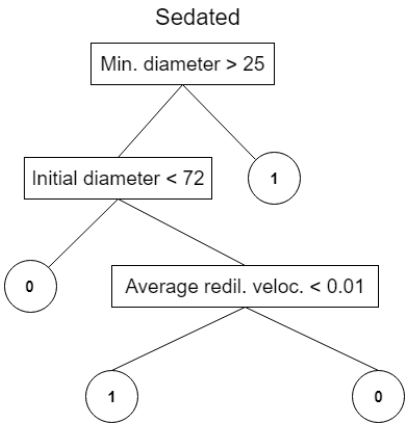


Figure 2: Decision tree fitted to the sedated group data. Leaf nodes containing 0 correspond to control group and nodes with values 1 correspond to the test group.

region) and the contraction time were shorter in the control group. These results are in accordance with previous studies mentioned in Section 2.

The analysis of the anesthetized animals showed that they cannot be divided into two classes reliably. It is assumed that while anesthesia can prevent stress and allows a convenient investigation of pupillary reactions for a longer period, it also diminishes the differences between the two groups in this autonomic response. The classifier achieved only 60% accuracy. The selected predictors were the amplitude of contraction, the average redilation speed and the time required to reach the maximal redilation speed.

4.2 Discussion

The results showed that anesthetized animals cannot be used in these experiments, so the attention was focused on the sedated animals. However, the sedated animals were still able to move their heads during the experiments, which affected negatively the recording of the pupil with the closely placed camera.

Based on the decision tree analysis, the initial diameter and the minimum diameter seemed to be the most reliable features for the classification, although the dynamic, time-related features seemed to be less relevant (or more samples (rats) are required to detect the potential differences).

From these observations, it could be concluded that a more robust measurement process was required, which had a relatively short period and provided multiple light stimuli to induce reoccurring reflex responses and minimum diameter occasions. thus enabling a complex and more detailed analysis.

5 Improved pupillometry

5.1 Improved data acquisition

To improve the robustness of the recordings, the measurement setup was redesigned. Around the camera lens, an IR LED (infrared light-emitting diode) ring was attached, setting the camera and the illuminating IR LEDs (nearly) on the same optical axis. With this setting, the camera is able to detect the light reflected from the retina, causing the so-called 'Bright pupil effect' [17], and the camera can be placed farther from the animal, thus tiny animal movements do not affect the recording quality. In Figure 3 the differences between the old and new setups are presented.

A new embedded system was also developed to schedule the experiments, which is controlled from a graphical user interface on the PC to initiate different visible light impulse sequences. This scheduler implements accurate time synchronization and manages the timing of light stimuli. The system uses an RGB LED, which allows us to produce different light intensities and colors.

With these improvements, the signal-to-noise ratio was enhanced (the pupil is better detectable) in these images and the recordings are more robust and precise. The improved image quality can be observed in Figure 3.b.

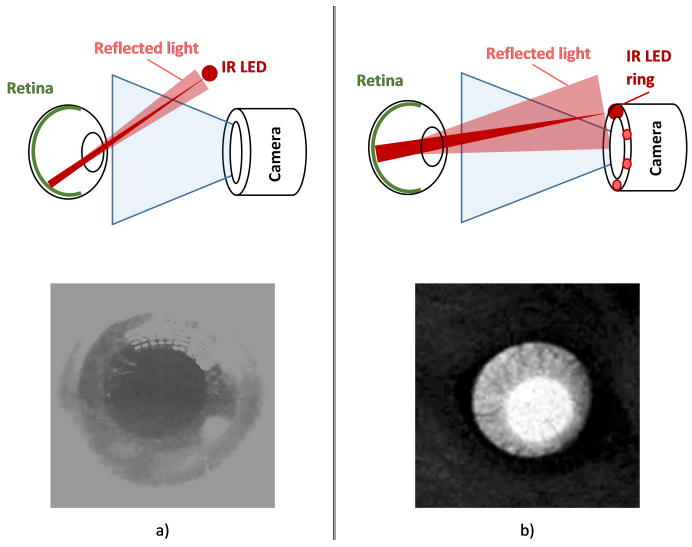


Figure 3: Difference between the old (a) and new (b) measurement setups. a) The camera is close to the eye and the illuminating infrared LED placed far from the camera's optical axis. The camera records black pupil region (reflected light never reaches the camera). b) The camera is placed farther from the eye and the illuminating LEDs are close to the optical axis. The camera records bright pupil region (reflected light reaches the camera).

5.2 Pupil segmentation dataset

The images recorded with the new measurement setup have a completely different nature, as it can be seen in Figure 3.b. Therefore, our previously developed method [10] cannot be used (different intensity levels, different resolution, etc.). To support the development of a new pupil segmentation algorithm and to validate the new setup, 56 experimental videos were recorded each containing more than 5000 frames. From these videos, 2564 randomly chosen frames were manually annotated (ellipses manually fitted to the pupil regions). An additional set of 329 images were selected and annotated to form a challenging test dataset.

Traditional image preprocessing methods were implemented to detect the eye region in the image, crop it and enhance the contrast. The preprocessed dataset contains 128×128 pixels-sized images and the corresponding binary pupil masks. Figure 4 shows some samples and the corresponding pupil masks from the pupil segmentation dataset.

The dataset is publicly available [9].

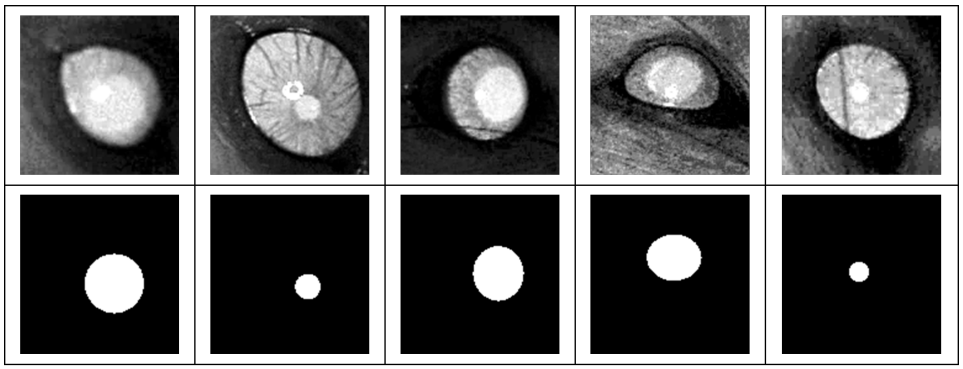


Figure 4: Example samples from the pupil segmentation dataset. Top row: original samples. Bottom row: corresponding pupil mask images.

6 Pupil segmentation

6.1 Pupil segmentation using neural network

A modern way to solve a segmentation problem is to use neural networks. Fully-convolutional networks are frequently used in segmentation problems as was explained in Section 2. In this paper, a U-shaped structure designed for biomedical applications, the U-Net structure [20] was used to perform the pupil segmentation task – the calculation of binary pupil masks. The training of the network required manually annotated data.

Our dataset contains input images with 128×128 resolutions. The originally published version on the U-Net worked with different input sizes, thus adaptations

were required. Three layers (each performs convolution and max-pooling) were applied and the number of channels was doubled after each pooling, as the original paper suggested. At the bottom of the U-shaped structure, 512 pieces of 16×16 sized feature maps were calculated. The abstract visualization of the used structure can be observed in Figure 5. Instead of random initialization, the "de-convolution" layers' weights were initialized to perform bilinear up-sampling. To compare the predicted and ground-truth pupil masks, binary cross-entropy was used as the loss function. Adam method [12] was used as the optimizer and the batch size was 64.

Data augmentation was employed to prevent model overfitting (random flips, random crop and resize). No other regularization techniques were used. The training process was optimized by the analysis of the training loss and validation loss curves. 10% of the training dataset served as the validation set. Early-stopping was applied to terminate the training process when the validation loss stopped decreasing. This setup was tested with different learning rates. The best performing model ran for 200 epochs, with the learning rate of 0.001. The algorithm was implemented in PyTorch [18].

The output of the neural network was an almost binary image. The pixel values were close to 1.0 when a pixel was considered as part of the pupil. The output was binarized with an empirically set threshold value. In the resulting binary images, contours were found and ellipses were fitted on these points. The final output was the mask of the fitted ellipse. If multiple ellipses were identified, the false detections were filtered out by simple rules based on the sizes, shapes, and locations of the ellipses. A set of example images and the predicted ellipses (without false detection filtering) are presented in Figure 6.

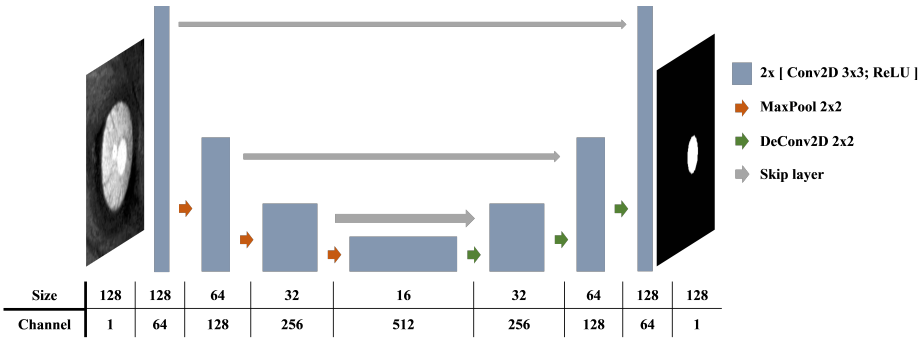


Figure 5: The neural network structure similar to [20].

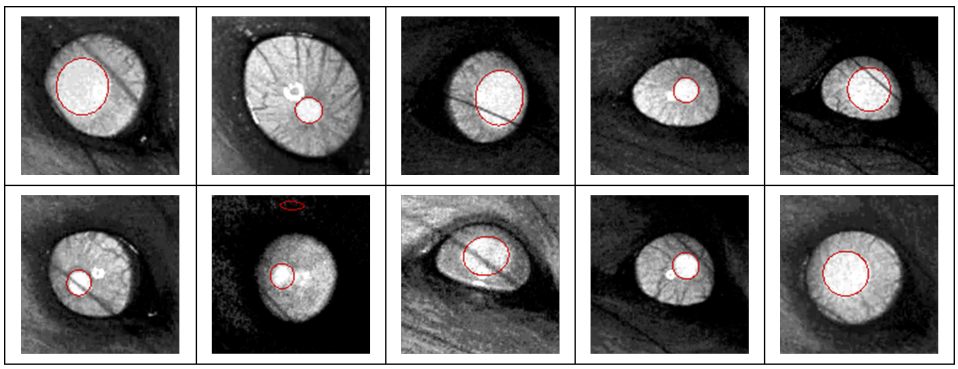


Figure 6: Example result images. Red ellipses present the predicted ellipses' borders.

6.2 Results

The trained model was evaluated on the test dataset containing 329 images. The Jaccard-index (Intersection of Union) was determined to compare the ground-truth and predicted pupil masks. The average Jaccard-index value on the test dataset was 0.815. This test set is challenging and there were samples, which were completely misclassified by the algorithm. Therefore the median value, which is more robust to these outliers was calculated too. The median relative error was 0.883.

Besides the pupil masks, the major axes lengths (diameters) of the annotated and predicted ellipses were also compared, which was important, because the original output of the system was the pupillogram, the curve of pupil diameters. The average relative diameter error was 12%, the median relative error was 4%. In Figure 7.a the sorted original and the corresponding predicted diameters are presented. It can be seen that most of the errors occurred when the pupils' diameters were smaller than 20 pixels. This information loss might be caused by the down-sampling section of the U-shaped structure. In Figure 7.b the relationship between the Jaccard-index and the relative diameter error can be observed. Small Jaccard-index value not necessarily implies considerable relative error. This can be explained because if only a small part of the mask is missing, this will reduce the Jaccard-index value (error of the masks). However, it is possible that the missing part only has an influence on the minor axis length of the ellipse, and the major axis length – the extracted diameter value – remains the same.

The diameter measurement accuracy can be improved with post-processing because additional filtering is available while the algorithm is being run on consecutive video frames. This time domain correlation could be utilized in more complex neural network structures too, which could accept a part of a video as their input. This possibility will be investigated in our future work.

Besides the objectively measured improvements, the pupillometry with the new measurement setup and segmentation algorithm will have a positive effect on the

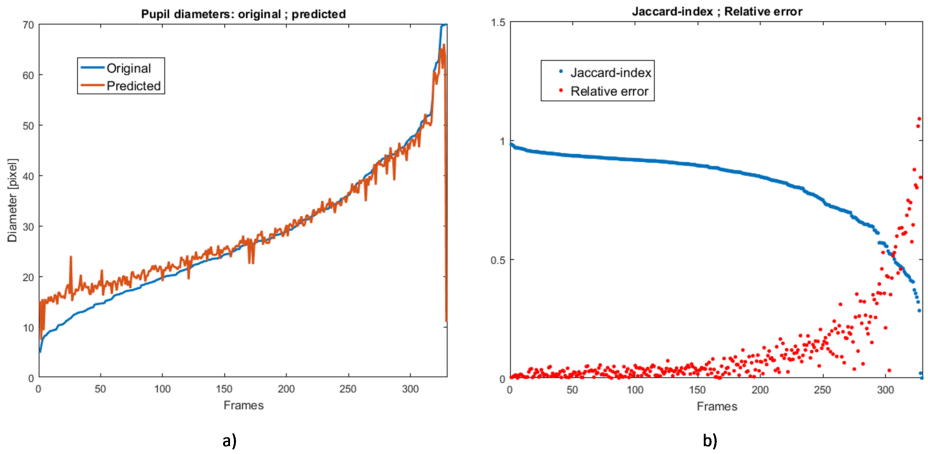


Figure 7: a) Sorted original diameters and the corresponding predicted diameters. b) Sorted Jaccard-indecies and the corresponding relative diameter errors.

classification of the animals too. As the former setup (one impulse/experiment) is inherited in the current measurements (impulse sequences), at least the same features can be extracted. However, the relations between the consecutive impulse responses will unfold more complex details about the alterations of the autonomic nervous system.

7 Concluding remarks

Pupillometry was applied to study the potential schizophrenia-like alterations in the PLR in WISKET rats. A 40-dimensional feature vector was assigned to each recorded video and the dataset was analyzed with decision tree-based classification. The results suggested that pupillary control showed significant alterations in WISKET rats, and the classification based on pupillometry might be applied as an additional examination during the breeding process. Based on the observations, the measurement process was redesigned to induce the "bright pupil effect", which offers more robustness to the experiments. To develop a new pupil segmentation algorithm, a publicly available pupil segmentation dataset was created. A fully-convolutional neural network was trained to produce binary pupil masks. On the test dataset, the relative pupil diameter predictor achieved low, 4% median error.

References

[1] Bär, Karl-Jürgen, Boettger, Michael Karl, Schulz, Steffen, Harzendorf, Christina, Agelink, Marcus Willy, Yeragani, Vikram K, Chokka, Prtap, and

Voss, Andreas. The interaction between pupil function and cardiovascular regulation in patients with acute schizophrenia. *Clinical Neurophysiology*, 119(10):2209–2213, 2008. DOI: 10.1016/j.clinph.2008.06.012.

[2] Bär, Karl-Jürgen, Koschke, Mandy, Boettger, Michael Karl, Berger, Sandy, Kabisch, Alexander, Sauer, Heinrich, Voss, Andreas, and Yeragani, Vikram K. Acute psychosis leads to increased qt variability in patients suffering from schizophrenia. *Schizophrenia research*, 95(1):115–123, 2007. DOI: 10.1016/j.schres.2007.05.034.

[3] Büki, Alexandra, Kalmár, György, Kekesi, Gabriella, Benedek, Gyorgy, Nyúl, László G., and Horvath, Gyongyi. Impaired pupillary control in "schizophrenia-like" WISKET rats. *Autonomic Neuroscience*, 213:34–42, 2018. DOI: 10.1016/j.autneu.2018.05.007.

[4] Fuhl, Wolfgang, Santini, Thiago, Kasneci, Gjergji, and Kasneci, Enkelejda. Pupilnet: convolutional neural networks for robust pupil detection. *arXiv preprint arXiv:1601.04902*, 2016.

[5] Fuhl, Wolfgang, Santini, Thiago, Kasneci, Gjergji, Rosenstiel, Wolfgang, and Kasneci, Enkelejda. Pupilnet v2. 0: Convolutional neural networks for cpu based real time robust pupil detection. *arXiv preprint arXiv:1711.00112*, 2017.

[6] Hermesh, Haggai, Shiloh, Roni, Epstein, Yoram, Manaim, Hillel, Weizman, Abraham, and Munitz, Hanan. Heat intolerance in patients with chronic schizophrenia maintained with antipsychotic drugs. *American Journal of Psychiatry*, 157(8):1327–1329, 2000. DOI: 10.1176/appi.ajp.157.8.1327.

[7] Horváth, Gyöngyi, Liszli, Péter, Kékesi, Gabriella, Büki, Alexandra, and Benedek, György. Characterization of exploratory activity and learning ability of healthy and 'schizophrenia-like' rats in a square corridor system (ambitus). *Physiology & behavior*, 169:155–164, 2017. DOI: 10.1016/j.physbeh.2016.11.039.

[8] Horvath, Gyongyi, Petrovszki, Zita, Kekesi, Gabriella, Tuboly, Gabor, Bodosi, Balazs, Horvath, Janos, Gombkötő, Peter, Benedek, Gyorgy, and Nagy, Attila. Electrophysiological alterations in a complex rat model of schizophrenia. *Behavioural brain research*, 307:65–72, 2016. DOI: 10.1016/j.bbr.2016.03.051.

[9] Kalmár, György, Büki, Alexandra, Kékesi, Gabriella, Horváth, Gyöngyi, and Nyúl, László G. Pupil segmentation dataset. Available at: https://github.com/Gyoorey/PupilDataset.

[10] Kalmár, György, Büki, Alexandra, Kékesi, Gabriella, Horváth, Gyöngyi, and Nyúl, László G. Image processing-based automatic pupillometry on infrared videos. *Acta Cybernetica*, 23(2):599–613, 2017. DOI: 10.14232/actacyb.23.2.2017.10.

[11] Kékesi, Gabriella, Petrovszki, Z, Benedek, György, and Horváth, Gyöngyi. Sex-specific alterations in behavioral and cognitive functions in a "three hit" animal model of schizophrenia. *Behavioural brain research*, 284:85–93, 2015. DOI: 10.1016/j.bbr.2015.02.015.

[12] Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] Lidsky, Arnold, Hakerem, Gad, and Sutton, Samuel. Pupillary reactions to single light pulses in psychiatric patients and normals. *Journal of Nervous and Mental Disease*, 1971. DOI: 10.1097/00005053-197110000-00007.

[14] Lu, Wei, Tan, Jinglu, Zhang, Keqing, and Lei, Bo. Computerized mouse pupil size measurement for pupillary light reflex analysis. *Computer Methods and Programs in Biomedicine*, 90(3):202–209, 2008. DOI: 10.1016/j.cmpb.2008.01.002.

[15] McLaughlin, Robert A. Randomized hough transform: better ellipse detection. In *TENCON'96. Proceedings., 1996 IEEE TENCON. Digital Signal Processing Applications*, volume 1, pages 409–414. IEEE, 1996. DOI: 10.1109/tencon.1996.608850.

[16] Mohan, Kabhilan, Harper, Matthew M, Kecova, Helga, Ye, Eun-Ah, Lazic, Tatjana, Sakaguchi, Donald S, Kardon, Randy H, and Grozdanic, Sinisa D. Characterization of structure and function of the mouse retina using pattern electroretinography, pupil light reflex, and optical coherence tomography. *Veterinary ophthalmology*, 15(s2):94–104, 2012. DOI: 10.1111/j.1463-5224.2012.01034.x.

[17] Morimoto, C.H, Koons, D, Amir, A, and Flickner, M. Pupil detection and tracking using multiple light sources. *Image and Vision Computing*, 18(4):331–335, 2000. DOI: 10.1016/s0262-8856(99)00053-0.

[18] Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca, and Lerer, Adam. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[19] Petrovszki, Zita, Adam, Gabor, Tuboly, Gabor, Kekesi, Gabriella, Benedek, Gyorgy, Keri, Szabolcs, and Horvath, Gyongyi. Characterization of gene–environment interactions by behavioral profiling of selectively bred rats: The effect of NMDA receptor inhibition and social isolation. *Behavioural brain research*, 240:134–145, 2013. DOI: 10.1016/j.bbr.2012.11.022.

[20] Ronneberger, O., P.Fischer, and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. DOI: 10.1007/978-3-319-24574-4_28.

[21] Yuen, HK, Illingworth, John, and Kittler, Josef. Detecting partially occluded ellipses using the Hough transform. *Image and Vision Computing*, 7(1):31–37, 1989. DOI: 10.1016/0262-8856(89)90017-6.

[22] Zahn, Theodore P and Pickar, David. Autonomic activity in relation to symptom ratings and reaction time in unmedicated patients with schizophrenia. *Schizophrenia research*, 79(2):257–270, 2005. DOI: 10.1016/j.schres.2005.05.025.

[23] Zhu, Danjie, Moore, Steven T, and Raphan, Theodore. Robust pupil center detection using a curvature algorithm. *Computer methods and programs in biomedicine*, 59(3):145–157, 1999. DOI: 10.1016/s0169-2607(98)00105-9.

Received 27th September 2018

Acta Cybernetica **24** (2019) 211–248.

The Logic of Aggregated Data

Tjalling Gelsema*

Abstract

A notion of generalization-specialization is introduced that is more expressive than the usual notion from, e.g., the UML or RDF-based languages. This notion is incorporated in a typed formal language for modeling aggregated data. Soundness with respect to a sets-and-functions semantics is shown subsequently. Finally, a notion of congruence is introduced. With it terms in the language that have identical semantics, i.e., synonyms, can be discovered. The resulting formal language is well-suited for capturing faithfully aggregated data in such a way that it can serve as the foundation for corporate metadata management in a statistical office.

Keywords: aggregation, information modeling, semantics

Introduction

This article is a sequel to [8]. The reader is therefore advised to make her- or himself acquainted of the notions and the results of [8]. Also, we inform the reader that this article is written from the perspective of information management in a statistical office. Its results however, we feel, are applicable in any situation where large quantities of aggregated datasets need to be managed faithfully.

One of the distinguishing features of a national statistical institute (NSI) is the large amount of information it harbors, dealing with many different social and economic phenomena. Managing this mass of information, i.e., making sure that the right pieces of information are available at any situation where they are required, is both necessary as well as nontrivial. For various reasons — accuracy, professionalism, transparency and coherence to name a few — an NSI should be aware of all the social and economic concepts it uses to produce statistics, across the entire process from data collection to publication.

While Statistics Netherlands (SN) is very keen on managing these concepts for the statistics that are published on its website, overall office-wide management of information across the statistical production process lags behind. Apart from the obvious, such as low risk of public exposure, there are various reasons why office-wide information management receives less attention and is difficult to achieve.

*Statistics Netherlands, Henri Faasdreef 312, 2492 JP Den Haag E-mail: te.gelsema@cbs.nl

DOI: 10.14232/actacyb.24.2.2019.4

First, true office-wide information management requires local investments, from the various departments the organization consists of, while the return on the investments is often less apparent from the point of view of these departments. The administration, i.e., the proper naming and describing, of the numerous variables, classifications, datasets, production rules, etcetera, is sometimes considered drudgery and mostly for the benefit of others.

Second, variables, classifications, statistical information models, micro data and aggregated data show dependencies that makes proper management of, say, variables through a variable management system, separately from that of, say, classifications through a classification management system, hard to realize. As an example of such a dependency, consider variables like *turnover generated from the sales of shoes*, *turnover generated from the sales of jeans*, etcetera. These variables are properly managed only if they are seen as one generic variable, *turnover generated from the sales of a good* say, that is 'indexed' by some classification of goods: only then need changes, e.g., in the definition of the variables, be recorded only once instead of for each of the indexed variables separately. For the converse, classifications that depend on variables, examples can be given as well.

In many organizations such as SN there is a need for *corporate metadata management*, which is aimed at integrating these separate initiatives in a meaningful way. However, uncovering dependencies that are meaningful from the perspective of corporate metadata management is harder than is acknowledged by initiatives such as the GSIM [22]. Besides, as a separate discipline, studying structural metadata, which is aimed at discovering these dependencies, has received too little attention in the scientific literature anyway (but see, e.g., [20]). In the sequel we will give some examples that fail to be handled properly by existing metadata frameworks. As we will see, having access to such dependencies can make various tasks much easier, but the systems or frameworks that should record them, we claim, are either lacking or are inadequate. As a result, the administrative tasks mentioned earlier are not properly supported and they therefore require more human effort than necessary.

SN has initiated a program to gain an office-wide overview of all of its datasets that are one 'steady state' behind its output tables in the statistical process: datasets that have reached a certain stability in the sense that they are no longer subject to changes, and are one step away from publishing. This is part of a larger effort to arrive at a complete overview, from data collection to publishing, which is carried out in the opposite direction (i.e., from publishing to data collection). During the years 2013 and 2014, these 'pre-output tables' were named and described both in general terms as well as in terms of the variables they consist of, in a joint effort that involved every department responsible for publishing statistics. The goal was to have office-wide access, by the beginning of 2015, to both the descriptions of these datasets (the so-called dataset designs) and through them authorized access to the datasets themselves. To achieve this, a separate department has been established, the Data Service Center (DSC), which is responsible for offering access to these data for the rest of the organization. The DSC also develops and manages the automated system for storing and retrieving datasets and dataset designs, of-

fers guidelines for naming and describing datasets and variables, and offers overall support to statisticians in applying these guidelines, to name a few of the DSC's responsibilities.

While the DSC has put much effort in what we refer to as the 'non-structural' business rules of dataset design, among which are the naming conventions, the 'structural' business rules have received much less attention. Among these 'structural' business rules are the ones that, given an information model, determine whether a group of variables can be reasonably put together to form a dataset, or that determine, given two variables, whether the one is an aggregated form of the other, as with *total turnover generated by a class of businesses* and *turnover of a business*. The result is that the automated system that the DSC provides is not sufficiently capable of responding to natural queries that arise when datasets are designed, which requires a more global and interdependent view of all information assets than the DSC is able to provide.

One example of such a query is: given an object type (such as one of the types *person*, *household*, or *business*) list all the currently available variables that apply to that type (such as *age* and *income*, *household composition* and *income*, and *turnover* and *profit*, respectively). Another example is: given a family of variables, such as *turnover generated from the sales of shoes* and *turnover generated from the sales of jeans* examined earlier, list the generic variable and the classification that they depend on. The first query is natural, because it stimulates the reuse of existing variables. It is difficult to respond to, because the DSC does not record object subtyping: e.g., the variable *age* (recorded as a variable of type *person*) will be missing from the list when all variables of type *student* are requested. The second query is natural, again for purposes of reuse: the description of the generic variable together with the descriptions of each of the categories in the classification are sufficient to understand each of the variables in the family. The automated system of the DSC cannot respond to the query though, since it does not record this kind of dependency between variables and classification systems. As a result, each variable in the family needs to be described separately, which is an example of the extra human effort that is required because of inadequate support from DSC's information systems.

In order to lay down useful dependencies, we claim it is natural and beneficiary to have a graph-like perception of corporate metadata. Then, we claim, it is equally natural to view the structure of a dataset as a particular subgraph. To illustrate what we mean, consider the dataset below that records the ages and the incomes of two partners in a marriage, as well as the duration of the marriage. (We assume that we list these for all marriages that existed on a particular date, say January 1 2016, in a particular residential area, say Delft. We also assume that *partner 1* and *partner 2* in the marriage, abbreviated by 'pa.1' and 'pa.2' respectively, can be identified according to some criterion.)

The peculiarity with this dataset is that, without further ado, the correct understanding of it depends on the correctness of the labels 'pa.1' and 'pa.2' assigned to the first four columns. These labels are easily switched and switching them probably gets unnoticed on first sight. So the dataset has some internal structure in the

age pa.1	income pa.1	age pa.2	income pa.2	duration
31	20.500	35	22.000	4
62	40.200	57	45.000	31
...	...	...	...	...

Table 1: Data about partners in a marriage

sense that the first and the second column of data need to be treated as a pair, as well as the third and fourth column. Other than assigning labels to columns, the DSC however has no means to formally record such information, and neither has the GSIM.

We propose that these useful dependencies, between columns in a dataset in the example above, can be adequately expressed in the form of the graph below, which should be viewed as an information model of a small part of the 'real world' a statistician might be interested in. To summarize what is depicted in Fig. 1,

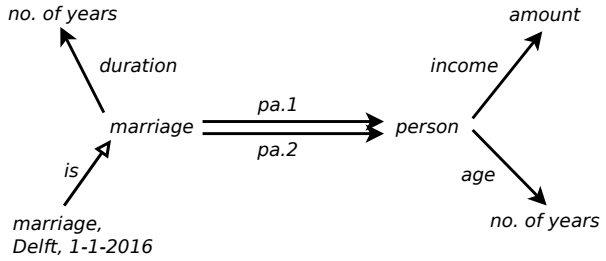


Figure 1: An information model

associated with a *marriage* are two *persons*, which are the partners involved in the marriage. Each person has characteristics *age* and *income*, expressed as a *no. of years* and an *amount*, respectively. A marriage has a characteristic *duration* which is also recorded as a *no. of years*. Each marriage recorded in Delft on January 1 2016 is a *marriage*.

The difference between what we call 'non-structural' and 'structural' business rules, respectively, is that only the latter can be expressed in a form sufficiently precise to be automatically (and correctly) interpreted by a computer program, in order to automatically respond to queries such as the ones above. While we feel that both 'non-structural' and 'structural' business rules are important in the development and use of automated systems that are successful in supporting information management in an NSI, studies of the latter are practically nonexistent in the scientific literature, at least when restricted to statistical information management or statistical metadata. There is however a huge source of literature available about techniques in formal semantics [23, 1], a subfield of theoretical computer science,

which, we feel, has valuable applications to statistical information management. This article is an attempt to show that these techniques can be successfully applied to solve the questions raised above.

In order to further delineate the scope of this article, we postulate that within the statistical process, there are two kinds of data transformations: those that change the structure of the data and those that keep the structure intact, but only change the contents of a dataset. Examples of the first kind are aggregation and row selection, examples of the second kind are data editing and imputation. We further postulate that, roughly, transformations of the first kind change or produce structural metadata, while transformations of the second kind change or produce quality metadata. Simply put, the second kind deals with changes of the *estimator*, while the first kind deals with changes of the *estimand*. This article only deals with changes of the *estimand* and hence only deals with changes in structural metadata, keeping quality metadata out of scope.

In [8] formal semantics, initial algebra semantics [9] and some category theory [13, 18, 3] in particular, were taken as a point of departure for developing 'structural' business rules for statistical data and metadata. The main ideas of [8] can be summarized as follows:

(i) The most natural and accurate interpretation of statistical data is the sets-and-functions interpretation. In this interpretation, variables, micro datasets, dimensional datasets, relations between object types (such as between a *marriage* and a *person* in Fig. 1) and also values[1] are seen as functions; object types and value types (such as *amount* in Fig. 1) are seen as sets. In this view for instance, a typical variable v takes an object (a person, a household, a business) from a set of objects p (the interpretation [8] gives to an object type) to a value taken from a set of values x (the interpretation of a value type). Hence we have the function $v : p \to x$;

(ii) To go from, e.g., variables and object type relations to a dataset and from a micro dataset to a dimensional dataset requires *operators* that act on sets and functions. For instance, in [8], column- and row-wise combining, functional composition and aggregation were considered as operators and were defined in the sets-and-functions interpretation;

(iii) Structural metadata should be 'aligned' in a natural way with these operators. This means that, for instance, combining variables column-wise in order to produce a dataset is 'mirrored' in combining descriptions of these variables in order to produce a description of the dataset. The mathematical consequence of this idea, using the initial algebra semantics point of departure, is that structural metadata are *terms*, built from the operators mentioned in (ii), that are mapped onto the sets and functions of (i) by means of a homomorphism.

In this article, terms that represent sets are called *types*; terms that represent functions are called *elements*. Elements have a *domain* and a *codomain*, which are types.

[1] We will see how values are interpreted as functions in the next section.

In the view of [8], the structural metadata for the dataset in Table 1 is the term

$$\langle \mathit{age} \circ \mathit{pa.1}, \mathit{income} \circ \mathit{pa.1}, \mathit{age} \circ \mathit{pa.2}, \mathit{income} \circ \mathit{pa.2}, \mathit{duration} \rangle \circ \mathit{in},$$

which is an element constructed using the operators column-wise combining $\langle \dots \rangle$ and functional composition $\circ$, that has the type

$$\mathit{marriage,\ Delft,\ 1\text{-}1\text{-}2016}$$

as a domain, and the type

$$\mathit{no.\ of\ years} \times \mathit{amount} \times \mathit{no.\ of\ years} \times \mathit{amount} \times \mathit{no.\ of\ years}$$

as a codomain. Note that the nodes that are incident with the directed edges from the graph in Fig. 1 are types that ensure the correct construction of the element above. For instance, functional composition $\circ$ is defined only for two elements (cf. the directed edges of Fig 1) of which the node incident on the tail (its domain) of the first equals the node incident on the head (its codomain) of the second. This implies that structural metadata expressed as an element, such as the one above, is safer than expressed through labels, which is the method that the DSC and the GSIM must resort to.

In addition, the 'structural' business rules that we need to answer the queries that we saw above require the following idea:

(iv) 'Structural' business rules for metadata are equivalences of terms, which, in the sets-and-functions interpretation of (i) become identities through the homomorphism of (iii). An example of such an identity is the fact that, under certain conditions, a two-stroke aggregation process can be reduced to a one-stroke aggregation step (see [8], Property (6)).

The contribution of this article to the ideas of [8] is twofold. First we extend the set of operators of [8] by a subtype operator and prove properties of it in the context of the other operators. Second, we prove that these properties — the 'structural' business rules — can be used to define a language for expressing structural metadata. For technical reasons explained below, this is more involved than it was for the set of operators of [8]. We explain these contributions briefly.

The idea of subtyping as an instrument of statistical information management, is that object types and their subtypes form a grouping mechanism for variables. A variable such as *age of a person* is recorded only once, viz. at the most generic type to which it applies: the object type *person*. Subtyping, e.g., the notion that any *student* is a *person*, allows access to that variable (e.g., *age of a student*) at more specific levels (*student*). This is the usual concept of inheritance; one of the foundations of the object-oriënted paradigm [15]. The open-headed generalization-specialization arrow of the Unified Modeling Language (UML, see [15]) is the usual way of recording subtyping; note that there is an occurrence of such an arrow in Fig. 1. On the other hand, more specific types (*student*) can give rise to variables (e.g., *year of application*) that do not apply at the generic level. Thus, subtyping

is an ordering of types that induces an ordering of groups of variables: the more generic the type, the less variables apply to it.

The notion of subtyping given in this article is more involved however than the usual notion from the UML or other ontology languages. In a(n automated) statistical process we usually need to know the justification for calling one type a subtype of another, in order to decide whether or not an object is a member of the subtype. Thus, we require that a subtype can only be defined — we prefer to say: *constructed* — out of a given type, if some selection criterion is supplied. This selection criterion can be a combination of a variable, applicable at the generic type, and a value in the range of that variable. For instance, we require that the construction of the object type *student* from *person* is supplied with a variable, say *is registered at a university?*, and with the value *yes* applicable to that variable.

The logical consequences of defining a subtype operator in this way are rather great, though. In [8], the universe of types could be defined independently of the universe of elements, which is a requirement for the use of equational logic [16] for defining the language as an initial algebra. With subtyping this is no longer the case, as the construction of a subtype depends on an element (e.g., the variable *is registered at a university?* in the example above). This means that, in a logical sense, there is an extra effort needed to untie the following knot:

a The universe of elements depends on the universe of types, as the domain and codomain of an element are both types;

b The universe of types depends on the universe of elements, by the subtype operator.

The article is organized as follows: in Sections 4 and 5 we define our language for structural metadata, from scratch, i.e., without support of any theory (such as equational logic) other than universal algebra [10]. After the Preliminaries of Section 1, in Sections 2 the subtype operator is introduced and its properties are shown there and in Section 3. To stress that Sections 2 and 3 are dealing with data, i.e., with sets and functions, we refer to the subtype operator there as *subset inclusion*. In Section 6 the semantics of the language is sketched and in Section 7 we give some examples that indicate the expressiveness of the language. We conclude with Section 8.

1 Preliminaries

We recall some of the notions of [8] and introduce some new ones.

For a set p, let Fp be the set of finite subsets of p. For sets x and y, let $x \times y$ be the binary Cartesian product of x and y, i.e., the set of all pairs $\langle d, e\rangle$ with $d \in x$ and $e \in y$. More generally, for any number $n > 1$, the Cartesian product of sets x_i, $i \leq n$, is denoted $x_1 \times \cdots \times x_n$ and consists of all n-tuples with elements taken from $x_1, \ldots, x_n$, respectively.

We let $1 = \{*\}$ be an arbitrary but fixed singleton set. We denote the empty set $\{\}$ by 0. Note that $x_1 \times \cdots \times x_n = 0$ if at least one of the x_i equals 0.

A commutative monoid is a structure $m = (m; +, 0)$ with m a set, $+$ an associative and commutative binary operation on m, and with $0 \in m$ an identity for $+$ (not to be confused with the empty set). More details can be found in [8]; the reader could consult [5] for full details. We refer to m simply as a monoid, since all monoids we consider here are commutative. For monoids m and $m' = (m'; +', 0')$, a function $h : m \to m'$ is a (monoid) homomorphism, if $h(a + b) = h(a) +' h(b)$ for all $a, b \in m$, and $h(0) = 0'$.

For a function $v : p \to x$, we call p the domain of v and x the codomain of v; the set of functions with domain p and codomain x is denoted by x^p. If v is considered to be a variable, then we say that v is defined on (the population) p and that v is defined for (the value set) x. All functions we consider in this article are total; for v as before this means that it associates with every e in p exactly one d in x, and then this d is denoted by $v(e)$. If, conversely, every d in x is associated with at most one e in p through $v(e) = d$, then v is an injection. The composition of $v : p \to x$ with $w : q \to p$ is the function $v \circ w : q \to x$ defined by $v \circ w(d) = v(w(d))$ for every $d \in q$. We will normally abbreviate $v \circ w$ by vw. The composition of two injections is an injection. The left and right identities for composition are the identity functions: if we let v as before, then $v \circ \mathsf{id}_p = v = \mathsf{id}_x \circ v$, where $\mathsf{id}_p : p \to p$ is defined by $\mathsf{id}_p(e) = e$ for every $e \in p$, and similarly for id_x.

The product (or: 'column-wise' combination) of n functions $v_i : p \to x_i$, $1 \leq i \leq n$ with $n > 1$, is defined in the following way: we let $\langle v_1, \ldots, v_n \rangle : p \to x_1 \times \cdots \times x_n$ be the function u defined by $u(e) = \langle v_1(e), \ldots, v_n(e) \rangle$ (i.e., an n-tuple) for all $e \in p$. Note that the product is defined for functions with a common domain only. Given x_i as above, we let $\pi_i^n : x_1 \times \cdots \times x_n \to x_i$, called the i-th projection, be the function that maps an n-tuple $\langle d_1, \ldots, d_n \rangle$ to the element d_i in x_i. Note that π_i^n is a family of functions: for every combination of $n > 1$ and i with $1 \leq i \leq n$, and every combination of sets $x_1, \ldots, x_n$, we assume that the i-th projection π_i^n applies as defined.

We stress that this article excludes the 'row-wise' combination of functions, an operation that was included in [8].

A function $w : p \to q$ is inverse-finite if $w^{-1}(d) = \{e \in p \mid w(e) = d\}$ is finite for every $d \in q$. The composition of two inverse-finite functions is inverse-finite and every injection is inverse-finite. For an inverse-finite w with domain and codomain as before, let $\delta(w) : q \to Fp$ be the function that maps an element $d \in q$ to the finite set $w^{-1}(d)$. Note that $\delta(w)$ is not an injection in general. For a function $v : p \to m$ with m a monoid, we let $\gamma(v) : Fp \to m$ be the function that maps a finite nonempty set $\{e_1, \ldots, e_k\}$ to $v(e_1) + \cdots + v(e_k)$, and the empty set 0 to the monoid identity 0. The mappings $\delta(w)$ and $\gamma(v)$ are called the dimensional structure induced by w and the elementary class parameter induced by v, respectively. Their composition $\gamma(v)\delta(w)$, called the aggregation of v by w, is denoted by $\alpha(v, w)$. Note that the composition 'factors through' Fp.

Aggregation, as defined by α, captures most of the more common aggregation operators, such as sums, maxima and minima, and (weighted) averages (see [8]). We conjecture however that medians are not covered by α.

For every $d \in x$ there is a unique function $\vec{d} : 1 \to x$ with $\vec{d}(*) = d$, and

conversely, every function $1 \to x$ 'picks out' a unique element in x. There is thus a one-to-one correspondence between x and the set of functions with domain 1 and codomain x. It is therefore natural, but not so common, to identify an element d in x with the function $\vec{d}$. Note that, for a function $v : x \to y$, we can then equally identify $v(d)$ with $v\vec{d}$ (since $v(d) = (v\vec{d})(*)$) and even with vd if we omit notational differences. To sum up: in the sequel the phrase "d is an element of x" can mean $d \in x$ or it can mean $d : 1 \to x$; it will always be clear from the context whichever applies.

For each set x there is exactly one function with domain x and codomain 1 (viz. every element in x has $*$ as its image) and we write 1_x to denote this function. Also, for each set x there is a unique function with domain 0 and codomain x and we denote this function by 0_x.

We adopt the following notational conventions: we use the letters p, q, r and $x, x_1, \ldots x_i, \ldots, x_k, y, z$ to denote sets, we use a, b, c, d, e to denote elements from these sets (potentially interpreted as functions with 1 as domain, as explained above), we use u, v, w to denote arbitrary functions, m, m' to denote monoids, and h, g to denote monoid homomorphisms.

The reader is encouraged to interpret a set denoted by p, q or r as a set of 'objects (or entities) of statistical interest'. This gives an informal meaning to the notion of an 'object type', such as the object type *person* or the object type *household*: informally an 'object e is of type p' if $e \in p$. Similarly, any of the sets $x, x_1, \ldots, x_k, y, z$ and m, m' should be interpreted as a set of 'values', i.e., as a rough interpretation of a 'value type'. However, within the context of this article, we make no mathematical distinction between 'values' and 'objects', except perhaps in the case of a 'value type' that supports aggregation: we require that such a type is a monoid. (We also advocate that a classification system is a Boolean algebra — see [7] — but that is outside the scope of this article.) Thus formally we do not make any distinction between a set of 'objects' and a set of 'values'. Hence in particular, the general Cartesian product is also defined for sets of 'objects', or for any combination of sets of 'objects' and sets of 'values', and the finite powerset operator F also applies to sets of 'values'.

Thus, when we write, e.g., $v : p \to x$ then v informally corresponds to the typical notion of a variable: a mapping that takes an object e of object type p to a value $v(e)$ of value type x. However, any mathematical rule to distinguish a variable from an arbitrary function is problematic: should we call a mapping of the form $p \to x \times y$ a variable or not? And what about a mapping of the form $p \times x \to y$? Also: is $p \times q$ an object type when both p and q are designated as object types? See [8] for a more conceptual discussion on these issues.

We proceed formally. Below we list the most important identities involving composition, product and aggregation, discovered and proven in [8]. We assume that h is a monoid homomorphism in Equation 5, that w is inverse-finite in Equations 4, 5 and 6, and that w is an injection in Equation 7. We urge the reader to draw a diagram of the situation for each of the identities, in which proper domains and codomains of the functions involved are depicted as vertices, and the functions are depicted as directed edges, in a way similar to Fig. 2, which sketches the situation

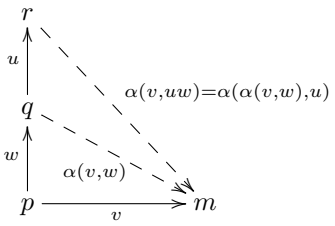


Figure 2: Distributivity of composition over aggregation

of Equation 4.

Associativity of composition:

$$(vw)u = v(wu) \tag{1}$$

Definition of projection:

$$\pi_1^2\langle v_1, v_2\rangle = v_1 \text{ and } \pi_2^2\langle v_1, v_2\rangle = v_2 \tag{2}$$

Distributivity of composition over product, right argument:

$$\langle vu, wu\rangle = \langle v, w\rangle \circ u \tag{3}$$

Distributivity of composition over aggregation, right argument:

$$\alpha(v, uw) = \alpha(\alpha(v, w), u) \tag{4}$$

Distributivity of composition over aggregation, left argument :

$$\alpha(hv, w) = h \circ \alpha(v, w) \tag{5}$$

Distributivity of product over aggregation, left argument:

$$\alpha(\langle v_1, v_2\rangle, w) = \langle \alpha(v_1, w), \alpha(v_2, w)\rangle \tag{6}$$

Cancellation law for aggregation:

$$\alpha(v, w)w = v \tag{7}$$

We stress that Equations 2, 3 and 6 can be easily extended to arbitrary products. We therefore also assume the correctness of

$$\pi_i^n\langle v_1, \ldots, v_n\rangle = v_i \tag{2'}$$

and

$$\langle v_1 u, \ldots, v_n u\rangle = \langle v_1, \ldots, v_n\rangle \circ u \tag{3'}$$

and

$$\alpha(\langle v_1, \ldots, v_n\rangle, w) = \langle \alpha(v_1, w), \ldots, \alpha(v_n, w)\rangle, \tag{6'}$$

respectively.

2 Subset inclusion

In this section we extend the constructs of the previous section with the mechanisms involved in forming a subset of a given set, given some conditions.

The principle motivation for extending the constructs of [8] with the formation of a subset is given by the observation that the theory developed in [8] lacks the means of relating a variable $u : p \to x$ with the variable $u' : p' \to x$ that is obtained from u by restricting its domain p to a subset $p' \subseteq p$. According to [8], u and u' can only be treated as separate variables, having separate and unrelated properties, which in general is not a desirable feature. To see this, take as an example of u the variable *age class of a person*, i.e., p is the set of persons and x is a set of age classes, such as $[16 - 25]$ and $[26 - 40]$. Then the variable *age class of a female person* is a variable $u' : p' \to x$, which is essentially just u only applied to the subset p' of women. It would be beneficiary if we could describe u' in terms of u, so that u' could inherit some of the properties of u, such as its definition. The first observation is thus that a subset induces variables that are derived from their 'principle form' in the sense that they are essentially the same, only restricted to this subset. It is the objective of this section to describe the mechanisms behind this derivation. As a prelude, note that the inclusion $i : p' \to p$ from p' into p given by $i(e) = e$ gives us the means to satisfactorily define u' as $u \circ i$.

An equally important observation is that the introduction of a subset p' of p gives rise to an asymmetry in the variables that are defined on p and p' respectively, in the sense that a variable $u : p \to x$ is 'equally defined' on p' (viz. through the inclusion i, as we saw above) but the reverse need not be true. Take for instance the variable *number of pregnancies carried to term* defined on the set p' of female persons: this variable makes no sense for the 'full' set p of persons. Thus, subsets, through inclusions, order the availability of variables: through an inclusion more, and more specialized, variables become available.

It is in this sense that the inclusion $i : p' \to p$ can be thought of as the generalization-specialization arrow of the class diagrams of the Unified Modeling Language (UML)[15]. There is however a crucial difference between our treatment of subsets sketched above and the UML generalization-specialization arrow: we only allow the creation of a subset p' from p if it is 'justified' by means of variables that are defined on p. To explain this, note that the set p' of female persons introduced above suggests that we can tell which entity of p is also an entity of p', viz. through the variable *sex*. Thus in the introduction of p' we tacitly used as a selection criterion a variable $v : p \to \{m, f\}$, which assigns a gender (either m or f) to each member of p, and we selected those members e from p for which $v(e) = f$. This suggests that we may write $p' = p_{(v,f)}$ or, more suggestively, $p' = p_{(v=f)}$. It is in this way that we require that each subset is to be constructed from the combination of a set, a variable and a constant, and we will generalize this to the combination of a set and two variables in a minute. Since the inclusion from a subset p' into p is given once p' is defined in this way, we may also write $i = i_{(v,f)}$ (or $i = i_{(v=f)}$).

By identifying an element d from a set x with the constant $d : 1 \to x$, as ex-

plained in the Preliminaries, we obtain the general situation for subset construction sketched in Fig. 3. Thus for a variable $v : p \to x$ and an element $d : 1 \to x$, we

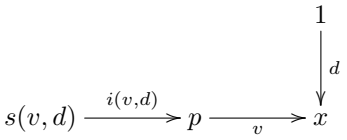


Figure 3: The subset induced by v and d

define $s(v, d)$ as the set $\{e \in p \mid v(e) = d(*)\}$. We also let $i(v, d)$ be the function $i : s(v, d) \to p$ defined by $i(e) = e$ for all $e \in s(v, d)$.

For technical reasons mainly, we immediately want to make the situation sketched in Fig. 3 even more general. First we observe that any constant $d : 1 \to x$ can be turned into a constant mapping $d' : p \to x$ (by which we mean that d' maps every element of p to the same element $d \in x$) viz. by composing it with the unique map $1_p : p \to 1$ defined in the Preliminaries. Thus, if we let $d' = d \circ 1_p$ then d' is essentially the same as d: both always yield the element $d \in x$. Note however that now both d' and v are variables defined on p. Thus the subset $s(v, d)$ defined above is identical to the set $\sigma(v, d') = \{e \in p \mid v(e) = d'(e)\}$, since $d'(e) = d(1_p(e)) = d(*)$. The introduction of σ however allows a more general construction in which both arguments are arbitrary variables v and w defined on p and for x, as sketched in Fig. 4. The case $w = d'$ then yields the special case of Fig 3. Thus, in the more

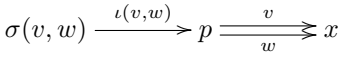


Figure 4: The subset induced by v and w

general situation sketched in Fig. 4, we let the *subset induced by v and w*, denoted by $\sigma(v, w)$, be defined as the set $\{e \in p \mid v(e) = w(e)\}$ and, as before, we let the *inclusion induced by v and w*, denoted by $\iota(v, w)$, be the function $\iota : \sigma(v, w) \to p$ defined by $\iota(e) = e$ for all $e \in \sigma(v, w)$. Clearly, $\iota(v, w)$ is an injection.

The definitions of σ and ι are inspired by category theory [3, 18, 13] and the notion of an equalizer in particular, which in a more abstract sense characterizes the subset and the inclusion induced by two functions. We give the definition involved, since it can help prove properties of σ and ι that we state in the next section, but we stress that understanding it is not essential for understanding the development of the theory in this article. Following the terminology of category theory, given objects p and x, and arrows $v, w : p \to x$, an *equalizer of v and w* is an object σ together with an arrow $\iota : \sigma \to p$ for which it holds that $v\iota = w\iota$, and such that for every object σ' and arrow $\iota' : \sigma' \to p$ with $v\iota' = w\iota'$, there is a unique arrow $\mu : \sigma' \to \sigma$ such that $\iota' = \iota\mu$.

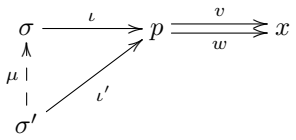


Figure 5: An equalizer of v and w

It can be shown, in the category of sets and functions, that $\sigma(v, w)$ and $\iota(v, w)$ form an equalizer. First, it is easy to see that

$$v\iota(v, w) = w\iota(v, w), \tag{8}$$

by taking an arbitrary element e from $\sigma(v, w)$. We remind the reader that $v\iota(v, w)$ is shorthand for $v \circ \iota(v, w)$ and similarly for $w\iota(v, w)$. Second, assume that $vi' = wi'$ for a function $i' : s' \to p$, i.e., for every $d \in s'$ we have $v(i'(d)) = w(i'(d))$. This means that $i'(d) \in \sigma(v, w)$ for every $d \in s'$. Hence the mapping $u : s' \to \sigma(v, w)$ with $u(d) = i'(d)$ for every $d \in s'$ is well-defined and satisfies $i' = \iota(v, w) \circ u$. Since $\iota(v, w)$ is an inclusion, u is the only such mapping.

Intuitively, a second motivation for introducing the more general subset construction of Fig. 4 by means of an equalizer is that there are situations for which comparing two variables, instead of one variable and one constant, could be useful. Take for instance two variables v and w that both measure the income of a person, but through different means, e.g., through a survey and a register say. Then to investigate the set of persons for which both variables yield the same value requires the equalizer of v and w.

In some situations, it could even be more useful to take the subset of p for which both variables yield a *different* result, which upon first glance is a subset construction that cannot be realized through σ and ι, or through s and i. If however we assume the set $b = \{true, false\}$ of the Boolean values and an inequality function $\neq : x \times x \to b$, then this subset (and the inclusion similarly) can be expressed as $s(\neq\langle v, w\rangle, true)$ where '*true*' is a constant $1 \to b$ and $\neq\langle v, w\rangle$ is the application of the inequality function to the product of v and w. But then of course, assuming equality $= : x \times x \to b$, a similar construction shows that σ and ι can be expressed in terms of the subset construction of Fig. 3. Hence, with some assumptions, the combination of s and i is equally expressive as σ and ι. Since we made less assumptions when viewing a subset as an equalizer (cf. Fig. 4) we take σ and ι as atomic and consider s and i as useful derivations.

One obvious difference is that for σ and ι the order of their arguments is of no importance, while for s and i it is required that their second argument is a constant. Thus we have

$$\sigma(v, w) = \sigma(w, v) \tag{9}$$

and

$$\iota(v, w) = \iota(w, v). \tag{10}$$

3 More properties of subset inclusion

In this section we continue to investigate properties of the construction of subsets and subset inclusions, especially in the context of the other operators mentioned in the Preliminaries, and explain their relevance for statistical practice.

We first study a number of situations in which expressions containing σ and ι can be simplified. Let p, x, v and w be as before. Consider an injection $u : x \to y$. Then we have

$$\sigma(uv, uw) = \sigma(v, w) \tag{11}$$

and

$$\iota(uv, uw) = \iota(v, w), \tag{12}$$

since $u(v(e)) = u(w(e))$ if and only if $v(e) = w(e)$, for all $e \in p$. Next, consider the situation in which a subset of $\sigma(v, w)$ is induced by v and w, i.e., using the same condition under which $\sigma(v, w)$ is formed. This is the situation in which, e.g., all females from a set of females are selected: this set should of course remain unchanged. More precisely, we have the situation sketched in Fig. 6. It can be

$$\sigma(v\iota(v,w), w\iota(v,w)) \xrightarrow{\iota(v\iota(v,w),w\iota(v,w))} \sigma(v,w) \xrightarrow{\iota(v,w)} p \underset{w}{\overset{v}{\rightrightarrows}} x$$

Figure 6: The subset of a subset, both 'induced' by v and w

shown that

$$\sigma(v\iota(v, w), w\iota(v, w)) = \sigma(v, w) \tag{13}$$

and

$$\iota(v\iota(v, w), w\iota(v, w)) = \mathsf{id}_{\sigma(v,w)}, \tag{14}$$

as expected. The conditions for the third simplification are sketched in Fig. 7. Note that d and e are constants, as defined in the Preliminaries: they are functions

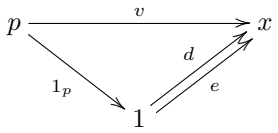


Figure 7: Subsets defined by constants d and e

$d, e : 1 \to x$ that are given by members d and e of x. Now suppose that $d \neq e$. Then we have

$$\sigma(v\iota(v, d1_p), e1_p\iota(v, d1_p)) = 0, \tag{15}$$

and

$$\iota(v\iota(v, d1_p), e1_p\iota(v, d1_p)) = 0_{\sigma(v,d1_p)}, \tag{16}$$

indicating that, e.g., selecting all males from a female population results in the empty set. Recall that the right hand side of Equation 16 is the unique map with the empty set as domain and $\sigma(v, d1_p)$ as codomain. Properties 15 and 16 are depicted in Fig. 8. We leave their proofs to the reader. Finally, the only subset of

$$\sigma(v\iota(v, d1_p), e1_p\iota(v, d1_p)) \xrightarrow{\iota(v\iota(v,d1_p),e1_p\iota(v,d1_p))} \sigma(v, d1_p) \xrightarrow{\iota(v,d1_p)} p$$

Figure 8: The subset of a subset, induced by different constants

the empty set is the empty set itself, so we have

$$\sigma(0_x, 0_x) = 0, \tag{17'}$$

where 0_x is the unique mapping $0 \to x$, and

$$\iota(0_x, 0_x) = 0_0, \tag{18'}$$

with 0_0 the unique mapping $0 \to 0$. More generally, if v is a mapping $p \to q$, then

$$\sigma(v, v) = p, \tag{17}$$

and

$$\iota(v, v) = \mathsf{id}(p). \tag{18}$$

Next consider the situation sketched in Fig. 9, where two subsets are formed using different pairs of variables as conditions. Take for example p as the set of persons, $\sigma(v_1, w_1)$ the subset of women, and $\sigma(v_2, w_2)$ the subset of elderly people (assuming, e.g., suitable conditions on the variables *sex* and *age*, respectively). The question is in what way these subsets are related. Intuitively, the two subsets are

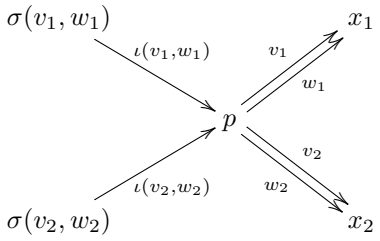


Figure 9: Two subsets induced by different pairs of variables

related through a third: the subset of elderly women. It should be clear that there are two ways of forming this subset: either by a restriction involving *age* on the subset of woman, or by a restriction involving *sex* on the subset of elderly, the results of which should be equal intuitively. Formally this situation is depicted in

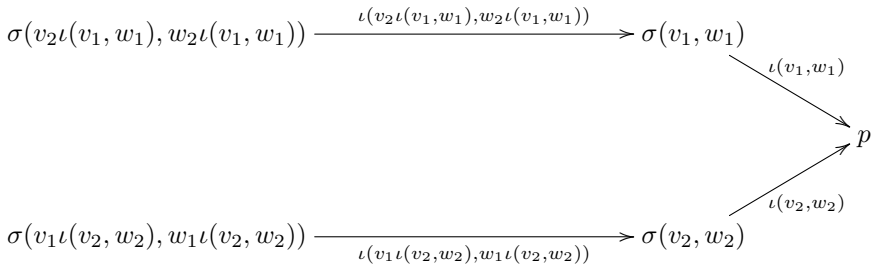


Figure 10: Two more subsets induced by different pairs of variables

Fig. 10. As expected the top-left subset equals the bottom-left, i.e., we have

$$\sigma(v_2\iota(v_1,w_1), w_2\iota(v_1,w_1)) = \sigma(v_1\iota(v_2,w_2), w_1\iota(v_2,w_2)). \tag{19}$$

To see this, note that the left-hand side reduces to

$$\{e \in \sigma(v_1,w_1) \mid v_2\iota(v_1,w_1)(e) = w_2\iota(v_1,w_1)(e)\},$$

which equals

$$\{e \in p \mid v_1(e) = w_1(e) \text{ and } v_2(e) = w_2(e)\},$$

since $e \in \sigma(v_1,w_1)$ if and only if $e \in p$ and $v_1(e) = w_1(e)$, and since $\iota(v_1,w_1)(e) = e$ for all $e \in \sigma(v_1,w_1)$. A similar reduction applies to the right-hand side of Equation 19. It should be equally clear that we then also have

$$\iota(v_1,w_1)\iota(v_2\iota(v_1,w_1), w_2\iota(v_1,w_1)) = \iota(v_2,w_2)\iota(v_1\iota(v_2,w_2), w_1\iota(v_2,w_2)), \tag{20}$$

following the paths of the arrows at the top and at the bottom of Fig. 10, respectively. This can be shown formally using the uniqueness condition of the equalizer construction of Fig. 5, the proof of which we leave to the reader.

The 'commutativity' laws specified in Equations 19 and 20 call for a change of notation: we let $\sigma(v_1{\sim}w_1)$ be an alternate notation for $\sigma(v_1,w_1)$ (and similarly for $\iota(v_1{\sim}v_2)$), we let $\sigma(v_1{\sim}w_1, v_2{\sim}w_2)$ be the left-hand side of Equation 19, and we let $\iota(v_1{\sim}w_1, v_2{\sim}w_2)$ be the left-hand side of Equation 20. This means that Equations 19 and 20 reduce to

$$\sigma(v_1{\sim}w_1, v_2{\sim}w_2) = \sigma(v_2{\sim}w_2, v_1{\sim}w_1) \tag{21}$$

and

$$\iota(v_1{\sim}w_1, v_2{\sim}w_2) = \iota(v_2{\sim}w_2, v_1{\sim}w_1), \tag{22}$$

respectively.

The subsets mentioned in Equation 21 as well as the inclusions mentioned in Equation 22 can also be formed in 'one stroke', viz. through the product operator. More precisely, in the situation of Fig. 9, we have that

$$\sigma(\langle v_1,v_2\rangle{\sim}\langle w_1,w_2\rangle) = \sigma(v_1{\sim}w_1, v_2{\sim}w_2), \tag{23}$$

and

$$\iota(\langle v_1, v_2\rangle{\sim}\langle w_1, w_2\rangle) = \iota(v_1{\sim}w_1, v_2{\sim}w_2), \tag{24}$$

the proof of which we leave to the reader.

The notation introduced just before Equation 21 can be extended to any number of (pairs of) arguments. We give the details by an inductive definition. Assume $v_j, w_j : p \to x_j$ with $1 \leq j \leq m$. Then we let $\sigma(v_1{\sim}w_1)$ be $\sigma(v_1, w_1)$ and $\iota(v_1{\sim}w_1)$ be $\iota(v_1, w_1)$ as before, and for $1 < j \leq m$ we let

$$\sigma(v_1{\sim}w_1, \ldots, v_m{\sim}w_m) = \sigma(v_m\iota_1^{m-1}, w_m\iota_1^{m-1}), \tag{25}$$

and

$$\iota(v_1{\sim}w_1, \ldots, v_m{\sim}w_m) = \iota_1^{m-1}\iota(v_k\iota_1^{m-1}, w_k\iota_1^{m-1}), \tag{26}$$

where ι_1^{m-1} abbreviates $\iota(v_1{\sim}w_1, \ldots, v_{m-1}{\sim}w_{m-1})$. Without proof, we claim that Equations 21, 22, 23, and 24 can be extended to any number of arguments. In particular, as far as Equations 21 and 22 are concerned, this means that

$$\sigma(v_1{\sim}w_1, \ldots, v_m{\sim}w_m) = \sigma(v_{\phi(1)}{\sim}w_{\phi(1)}, \ldots, v_{\phi(m)}{\sim}w_{\phi(m)}), \tag{27}$$

and

$$\iota(v_1{\sim}w_1, \ldots, v_m{\sim}w_m) = \iota(v_{\phi(1)}{\sim}w_{\phi(1)}, \ldots, v_{\phi(m)}{\sim}w_{\phi(m)}), \tag{28}$$

for any permutation ϕ of $\{1, \ldots, m\}$.

We close this section by the observation that in some circumstances forming a subset in the context of aggregation can be simplified. Consider the situation sketched in Fig. 11, where we assume that x is a monoid and w is inverse-finite. To explain this situation, let p be a set of persons, let q be a set of households,

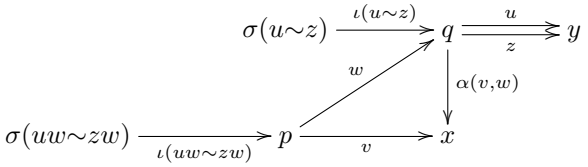


Figure 11: Subsets in the context of aggregation I

let w associate a person in p to the household in q he or she is a member of, and let v be the variable *income of a person*. Then, assuming $+$ is the monoid operation of x, $\alpha(v, w)$ is the *income of a household*, summing over the incomes of each member in a household. Now let $\iota(u{\sim}z)$ select the two-person households in q, where we assume that y is a set of household composition classes and u and z are suitable variables (or a suitable combination of a constant and a variable, as explained earlier). This means that $\iota(uw{\sim}zw)$ selects all the members of two-person households. The question is: how do $\alpha(v, w)$ and the *income of a two-person household* formed by $\alpha(v\iota(uw{\sim}zw), w\iota(uw{\sim}zw))$ relate? Intuitively, they should

be equal, as far as two-person households are concerned. We show that indeed we have

$$\alpha(v\iota(uw{\sim}zw), w\iota(uw{\sim}zw))\iota(u{\sim}z) = \alpha(v, w)\iota(u{\sim}z). \tag{29}$$

Let $d \in \sigma(u{\sim}z)$, i.e., we have $u(d) = z(d)$. We show that $\alpha(v, w)$ applied to d equals $\alpha(v\iota(uw{\sim}zw), w\iota(uw{\sim}zw))$ applied to d. Since these expand to

$$\sum_{d=w(e)} v(e) \quad \text{and} \quad \sum_{d=w\iota(uw{\sim}zw)(e)} v\iota(uw{\sim}zw)(e),$$

respectively, it suffices to show that $e \in \sigma(uw{\sim}zw)$ if and only if $d = w(e)$, since we then have $\iota(uw{\sim}zw)(e) = e$ as required. To show $e \in \sigma(uw{\sim}zw)$ whenever $d = w(e)$ is easy, since $d = w(e)$ implies that $u(w(e)) = z(w(e))$. The other implication is immediate and left to the reader.

It might be tempting to simplify Equation 29 by instead trying to prove the following equation

$$\delta(w\iota(uw{\sim}zw))\iota(u{\sim}z) = \delta(w)\iota(u{\sim}z).$$

This fails however since the codomains of both sides differ.

Finally, a similar but simpler case of forming subsets in the context of aggregation is sketched below (and we assume similar restrictions on x and w). In this

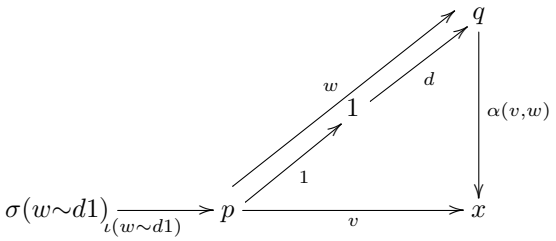


Figure 12: Subsets in the context of aggregation II

case we have

$$\alpha(v, w) \circ d = \alpha(v\iota(w{\sim}d1), w\iota(w{\sim}d1)) \circ d. \tag{30}$$

The proof is left to the reader, but we provide some intuition of the situation above: let p be a set of persons, w be the gender of a person (q is the set of the two sexes) and let v be an arbitrary variable, *income* say. Then the total income of men can be computed by first computing the totals for both men and woman followed by selecting the total for men (the left hand side of Equation 30), or men are selected first from the total popuation p and then their total is computed (the right hand side of Equation 30). Note that while the left hand side of Equation 30 is more concise and easier to understand, its right hand side is probably more computationally efficient.

4 Types and elements

We begin this section by pointing out that there is a crucial difference between the operators recalled in the Preliminaries and the subset operator σ introduced in Section 2. We note that all operators in the Preliminaries that produce a set, viz. the set Fp of finite subsets of p and the general Cartesian product $x_1 \times \cdots \times x_n$ of sets x_i, depend only on sets in their arguments: p and the x_i, respectively. Also, most operators that produce a function, viz. the composition $v \circ w$, the general product $\langle v_1, \ldots, v_n \rangle$, and the constructs $\delta(w)$ and $\gamma(v)$ for defining aggregation, depend only on functions in their arguments — the exceptions are the general projections π_i^n. Nevertheless, in short we have that sets depend on sets only, and functions depend on functions mostly.

The operator $\sigma(v, w)$ that produces a set in contrast relies on functions v and w in its arguments. This means that if we identify a set with a 'type', as we will do in this section, then $\sigma(v, w)$ is a so-called *dependent type* [21, 19]: one that needs additional values, or 'elements' as we will call them, in its construction. In the case of $\sigma(v, w)$, these values v and w are both 'elements' of the functional 'type' x^p, as expressed by the declaration $v, w : p \to x$.

In any case, the system of operators that was considered in [8] made life easy: it allowed for a language that could be defined within the framework of equational logic [16] and for which semantics was immediate ("zap", according to [9]). It made use of the fact that 'types' (or rather: sorts, as they are unfortunately called in the context of algebras) could be constructed independently of values or 'elements'. With the introduction of σ, the resulting system of operators does not have that advantage anymore.

Though there are extensions of initial algebra semantics that allow dependent types [14], how we choose to proceed is to introduce 'types' and 'elements' from scratch, i.e., without the use of some underlying theory other than universal algebra [10]. This is in part because our atomic formulae also deal with modalities (introduced in Definition 3 below) that are not treated by such extensions. Second, and more importantly, we think that the dependency between a typing relation and a congruence, as formulated in [14] Definition 3.5, is insufficient for our purposes and requires the inductive approach taken in this section, motivated by Example 1 below and embodied by Definition 6.

In this section we will give a number of elementary definitions that are used in the next section where they will be put together to form our language. First, both types and elements are certain terms, i.e., sequences of symbols formed through syntactic rules, the universes of which we will define below through mutual recursion, i.e., simultaneously. Then we will define type assignment, i.e., a relation between an element and its type(s). We note that types assigned to elements restrict the use of operators that apply to elements, as for instance the product of two elements requires that they have the same type as a domain. In this way we will limit the universes of types and elements to so-called well-formed types and elements. Similar restrictions will give us so-called well-defined types and elements: these will make sure that, for instance, the application of δ is limited to inverse-

finite elements only. Finally, we will define the concept of a congruence relation on types and elements and we show how the typing relation can be extended with it in a natural way: if an element is of a type t, then it should be of any type congruent with t. We note that, in turn, this gives rise to an extension of the universes of well-formed and well-defined types and elements.

We start our development with the simultaneous definition of types and elements, informally at first.

We assume a countable set A of *basic type symbols*. We also assume a countable set B of *basic element symbols* disjoint with A. The sets $T(A,B)$ and $E(A,B)$ of type terms (or just: types) and element terms (or just: elements) respectively, are given informally by the following mutually recursive grammars: a term p is a type if it is produced by the following grammar

$$\begin{aligned}\mathsf{p} \quad ::= \quad & \mathsf{a} \mid \mathsf{0} \mid \mathsf{1} \mid [\mathsf{p} \to \mathsf{q}] \mid \mathsf{p}_1 \times \cdots \times \mathsf{p}_n \mid \\ & \mathsf{F}(\mathsf{p}) \mid \boldsymbol{\sigma}(\mathsf{v}_1 {\sim} \mathsf{w}_1, \ldots, \mathsf{v}_m {\sim} \mathsf{w}_m)\end{aligned}$$

with $\mathsf{a} \in A$, $\mathsf{p}, \mathsf{p}_i, \mathsf{q} \in T(A,B)$, $\mathsf{v}_j, \mathsf{w}_j \in E(A,B)$, $n > 1$ and $m > 0$, with $i \leq n$ and $j \leq m$. A term v is an element if it is produced by

$$\begin{aligned}\mathsf{v} \quad ::= \quad & \mathsf{b} \mid \mathsf{0}(\mathsf{p}) \mid \mathsf{1}(\mathsf{p}) \mid \mathsf{id}(\mathsf{p}) \mid \mathsf{v} \circ \mathsf{w} \mid \langle \mathsf{u}_1, \ldots, \mathsf{u}_n \rangle \mid \\ & \boldsymbol{\pi}_i(\mathsf{p}_1, \ldots, \mathsf{p}_n) \mid \boldsymbol{\gamma}(\mathsf{v}) \mid \boldsymbol{\delta}(\mathsf{w}) \mid \boldsymbol{\iota}(\mathsf{v}_1 {\sim} \mathsf{w}_1, \ldots, \mathsf{v}_m {\sim} \mathsf{w}_m)\end{aligned}$$

with $\mathsf{b} \in B$, $\mathsf{p}, \mathsf{p}_i \in T(A,B)$, $\mathsf{v}, \mathsf{w}, \mathsf{v}_j, \mathsf{w}_j, \mathsf{u}_i \in E(A,B)$, $n > 1$ and $m > 0$, with $i \leq n$ and $j \leq m$.

When A and B are clear from the context, we abbreviate $T(A,B)$ by T and $E(A,B)$ by E.

In addition, we let the set $T(A)$ of elementary types be as follows: a term p is an elementary type if it is produced by

$$\mathsf{p} \quad ::= \quad \mathsf{a} \mid \mathsf{1} \mid [\mathsf{p} \to \mathsf{q}] \mid \mathsf{p}_1 \times \cdots \times \mathsf{p}_n \mid \mathsf{F}(\mathsf{p})$$

with $\mathsf{a} \in A$ and $\mathsf{p}, \mathsf{q}, \mathsf{p}_i \in T(A)$. Note that $\mathsf{0}$ is not an elementary type and that an elementary type does not depend on any elements.

The types $\mathsf{0}$ and $\mathsf{1}$ are called *zero* and *one*, respectively. The type $[\mathsf{p} \to \mathsf{q}]$ is the *function type induced by* p *and* q. The type $\mathsf{p}_1 \times \cdots \times \mathsf{p}_n$ is the *product type induced by* p_i. We stress that the product type is a family of constructs (one for every $n > 1$) and that the ellipsis $(\cdots)$ is not part of the type, but rather part of the metalanguage that is used to define the grammar. Thus $\mathsf{p} \times \mathsf{q}$ and $\mathsf{p} \times \mathsf{q} \times \mathsf{r}$ are types (provided p, q and r are types) and $\mathsf{p} \times \cdots \times \mathsf{q}$ is not a type. The type $\mathsf{F}(\mathsf{p})$ is the *finite power type induced by* p. For elements $\mathsf{v}_j, \mathsf{w}_j \in E$, the type $\boldsymbol{\sigma}(\mathsf{v}_1 {\sim} \mathsf{w}_1, \ldots, \mathsf{v}_m {\sim} \mathsf{w}_m)$ is the *subtype induced by* v_j *and* w_j. Again, this is a family of constructs (one for each $m > 0$) and the ellipsis is not part of the type.

Note that $\mathsf{0}(\mathsf{p})$ and $\mathsf{1}(\mathsf{p})$ define distinct elements for every $\mathsf{p} \in T$; each is called *zero* and *one*, respectively. When p is clear from the context (and when there is no danger of confusing them with their type counterparts) $\mathsf{0}(\mathsf{p})$ is sometimes abbreviated by $\mathsf{0}$ and $\mathsf{1}(\mathsf{p})$ is sometimes abbreviated by $\mathsf{1}$. The element $\mathsf{id}(\mathsf{p})$ is

the *identity on* p and we have such an element for every type p. We sometimes abbreviate $\mathsf{id}(\mathsf{p})$ by id. The names of the rest of the elements follow the names of their functional counterparts defined in Sections 1 and 2. So the element $\mathsf{v} \circ \mathsf{w}$ is called the *composition of* v *and* w and is sometimes abbreviated by vw. The ellipsis in $\langle \mathsf{u}_1, \ldots, \mathsf{u}_n \rangle$ is not part of the element, but part of the metalanguage. So $\langle \mathsf{u}, \mathsf{w} \rangle$ and $\langle \mathsf{u}, \mathsf{w}, \mathsf{v} \rangle$ are elements (provided u, w and v are elements) and $\langle \mathsf{u}, \ldots, \mathsf{w} \rangle$ is not. Also, for every $n > 1$ with $1 \leq i \leq n$ and every $\mathsf{p}_i \in T$, each $\boldsymbol{\pi}_i(\mathsf{p}_1, \ldots, \mathsf{p}_n)$ is a distinct element. When n and p_i are clear from the context, we abbreviate projection by $\boldsymbol{\pi}_i$. We use $\boldsymbol{\alpha}(\mathsf{v}, \mathsf{w})$ as an alternative notation for $\boldsymbol{\gamma}(\mathsf{v}) \circ \boldsymbol{\delta}(\mathsf{w})$. We note that $\boldsymbol{\iota}(\mathsf{v}_1 \sim \mathsf{w}_1, \ldots, \mathsf{v}_m \sim \mathsf{w}_m)$ is a family of constructs, one for each $m > 0$.

Formally now:

Definition 1. *Given a countable set A of basic type symbols and a countable set B of basic element symbols disjoint with A, the sets $T(A, B)$ and $E(A, B)$ of* types *and* elements *respectively are defined as*

$$T(A, B) = \bigcup_{k \geq 0} T_k \text{ and } E(A, B) = \bigcup_{k \geq 0} E_k,$$

where T_k and E_k are defined recursively as

$$\begin{aligned}
T_0 &= A \cup \{\mathsf{0}, \mathsf{1}\}, \\
E_0 &= B, \text{ and} \\
T_k &= T_{k-1} \cup \{[\mathsf{p} \to \mathsf{q}],\ \mathsf{p}_1 \times \cdots \times \mathsf{p}_n, \\
&\qquad \mathsf{F}(\mathsf{p}),\ \boldsymbol{\sigma}(\mathsf{v}_1 \sim \mathsf{w}_1, \ldots, \mathsf{v}_m \sim \mathsf{w}_m) \mid \\
&\qquad \mathsf{p}, \mathsf{q}, \mathsf{p}_i \in T_{k-1},\ \mathsf{v}_j, \mathsf{w}_j \in E_{k-1},\ n > 1 \text{ and } m > 0, \\
&\qquad \text{with } i \leq n \text{ and } j \leq m\}, \text{ and} \\
E_k &= E_{k-1} \cup \{\mathsf{0}(\mathsf{p}),\ \mathsf{1}(\mathsf{p}),\ \mathsf{id}(\mathsf{p}),\ \mathsf{v} \circ \mathsf{w},\ \langle \mathsf{u}_1, \ldots, \mathsf{u}_n \rangle, \\
&\qquad \boldsymbol{\pi}_i(\mathsf{p}_1, \ldots, \mathsf{p}_n),\ \boldsymbol{\gamma}(\mathsf{v}),\ \boldsymbol{\delta}(\mathsf{w}),\ \boldsymbol{\iota}(\mathsf{v}_1 \sim \mathsf{w}_1, \ldots, \mathsf{v}_m \sim \mathsf{w}_m) \mid \\
&\qquad \mathsf{p}, \mathsf{p}_i \in T_{k-1},\ \mathsf{v}, \mathsf{w}, \mathsf{v}_j, \mathsf{w}_j, \mathsf{u}_i \in E_{k-1},\ n > 1 \text{ and } m > 0, \\
&\qquad \text{with } i \leq n \text{ and } j \leq m\},
\end{aligned}$$

for all $k > 0$. We let $TE(A, B)$ be the set of all types and elements, i.e., $TE(A, B) = T(A, B) \cup E(A, B)$.

The set $T(A)$ of elementary types *is defined as*

$$T(A) = \bigcup_{k \geq 0} T'_k,$$

with T'_k defined recursively as

$$\begin{aligned}
T'_0 &= A \cup \{\mathsf{1}\}, \text{ and} \\
T'_k &= T'_{k-1} \cup \{[\mathsf{p} \to \mathsf{q}],\ \mathsf{p}_1 \times \cdots \times \mathsf{p}_n, \\
&\qquad \mathsf{F}(\mathsf{p}) \mid \mathsf{p}, \mathsf{q}, \mathsf{p}_i \in T'_{k-1} \text{ and } n > 1\}
\end{aligned}$$

for all $k > 0$.

Note that Definition 1 makes sense since $T_{k-1} \subseteq T_k$, $E_{k-1} \subseteq E_k$ and $T'_{k-1} \subseteq T'_k$ for all $k > 0$. Note also that $T(A) \subseteq T(A, B)$.

Given a term (an element or a type) $\mathsf{y} \in TE(A, B)$, the notions of a *subterm of* y and a *proper subterm of* y (i.e., a subterm of y not equal to y) are defined in the usual way, i.e., by induction on the structure of y according to Definition 1 above. We denote by $\mathsf{y}' \leq \mathsf{y}$ that y' is a subterm of y, and by $\mathsf{y}' < \mathsf{y}$ that y' is a proper subterm of y. Note that a type can be a subterm of an element and vice versa, due to, e.g., the projection construct and the subtype construct, respectively.

We stress that the elements in Definition 1 are untyped. This means that we do not yet have a relation between an element and a type that prevents constructing elements that make no sense. In other words, Definition 1 introduces elements and types that are intuitively incorrect. An example is the element $\mathsf{0(1)} \circ \mathsf{1(1)}$: intuitively, its subterms $\mathsf{0(1)}$ and $\mathsf{1(1)}$ represent the (unique) functions of type $[\mathsf{0} \to \mathsf{1}]$ and $[\mathsf{1} \to \mathsf{1}]$ respectively. However, the domain $\mathsf{0}$ of the first is incompatible with the codomain $\mathsf{1}$ of the second, which is required for a correct composition. We will correct this in a minute, when we introduce well-formed types and elements. Also, we note that Definition 1 introduces function types that represent the empty set, such as the type $[\mathsf{p} \to \mathsf{0}]$: if p represents a nonempty set, then no element should have type $[\mathsf{p} \to \mathsf{0}]$. More generally, since $\sigma(v{\sim}w)$ might yield the empty set, we have to be careful about assigning an element to a type of the form $[\mathsf{p} \to \boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{w})]$. The elementary types are 'safe' in this respect: if no basic type represents the empty set, then no elementary type represents the empty set.

The general concept of a typing relation that assigns one or more types to an element is given next. Elements that receive a type in this way are called well-formed, and types that are built from well-formed elements are well-formed. For reasons explained earlier in this section, we base a typing relation on an equivalence relation, such that elements are assigned to equivalent types, and equivalent elements receive identical types. Finally, we assume that basic element symbols are assigned a 'safe' type, through a given mapping.

Definition 2. *Let A and B be a set of basic type symbols and a set of basic element symbols respectively. Let $t : B \to T(A, B)$ be a mapping such that $t(\mathsf{b}) = [\mathsf{p} \to \mathsf{q}]$ with $\mathsf{q} \in T(A)$. Let $\equiv$ be an equivalence relation on types and elements; more specifically, let $\equiv \subseteq T(A, B)^2 \cup E(A, B)^2$. The* typing relation induced by t and $\equiv$, *denoted by $::_{t,\equiv} \subseteq E(A, B) \times T(A, B)$, is defined as the smallest relation such that the following conditions hold:*

(i) if $t(\mathsf{b}) = \mathsf{s}$ and s is well-formed, then $\mathsf{b} :: \mathsf{s}$,

(ii) if p is well-formed, then $\mathsf{0(p)} :: [\mathsf{0} \to \mathsf{p}]$, $\mathsf{1(p)} :: [\mathsf{p} \to \mathsf{1}]$ and $\mathsf{id(p)} :: [\mathsf{p} \to \mathsf{p}]$,

(iii) if $\mathsf{v} :: [\mathsf{q} \to \mathsf{r}]$ and $\mathsf{w} :: [\mathsf{p} \to \mathsf{q}]$, then $\mathsf{v} \circ \mathsf{w} :: [\mathsf{p} \to \mathsf{r}]$,

(iv) if $\mathsf{u}_i :: [\mathsf{p} \to \mathsf{p}_i]$, $1 \leq i \leq n$, then $\langle \mathsf{u}_1, \ldots, \mathsf{u}_n \rangle :: [\mathsf{p} \to (\mathsf{p}_1 \times \cdots \times \mathsf{p}_n)]$,

(v) if p_i is well-formed for every i with $1 \leq i \leq n$, then $\boldsymbol{\pi}_i(\mathsf{p}_1, \ldots, \mathsf{p}_n) :: [(\mathsf{p}_1 \times \cdots \times \mathsf{p}_n) \to \mathsf{p}_i]$,

(vi) if $\mathsf{v} :: [\mathsf{p} \to \mathsf{q}]$, then $\boldsymbol{\gamma}(\mathsf{v}) :: [\mathsf{F(p)} \to \mathsf{q}]$,

(vii) *if* $\mathsf{w} :: [\mathsf{p} \to \mathsf{r}]$, *then* $\boldsymbol{\delta}(\mathsf{w}) :: [\mathsf{r} \to \mathsf{F}(\mathsf{p})]$,

(viii) *if* $\mathsf{v}_j, \mathsf{w}_j :: [\mathsf{p} \to \mathsf{q}_j]$, $1 \leq j \leq m$,
then $\boldsymbol{\iota}(\mathsf{v}_1 \sim \mathsf{w}_1, \ldots, \mathsf{v}_m \sim \mathsf{w}_m) :: [\boldsymbol{\sigma}(\mathsf{v}_1 \sim \mathsf{w}_1, \ldots, \mathsf{v}_m \sim \mathsf{w}_m) \to \mathsf{p}]$,

(ix) *if* $\mathsf{v} :: \mathsf{s}$, $\mathsf{s} \equiv \mathsf{s}'$ *and* s' *is well-formed, then* $\mathsf{v} :: \mathsf{s}'$, *and*

(x) *if* $\mathsf{v} :: \mathsf{s}$, $\mathsf{w} \equiv \mathsf{v}$ *and* w *is well-formed, then* $\mathsf{w} :: \mathsf{s}$,

where in (i), $\mathsf{b} :: \mathsf{s}$ *is a shorthand notation for* $(\mathsf{b}, \mathsf{s}) \in ::_{t,\equiv}$ *and similarly for (ii) — (x), and where the set of* well-formed *types is the smallest set such that*

(xi) *every* $\mathsf{a} \in A$ *is well-formed, and* $\mathsf{0}$ *and* $\mathsf{1}$ *are well-formed,*

(xii) *if* p *and* q *are well-formed, then* $[\mathsf{p} \to \mathsf{q}]$ *is well-formed,*

(xiii) *if* p_i *is well-formed for every* i, $1 \leq i \leq n$, *then* $\mathsf{p}_1 \times \cdots \times \mathsf{p}_n$ *is well-formed,*

(xiv) *if* p *is well-formed, then* $\mathsf{F}(\mathsf{p})$ *is well-formed, and*

(xv) *if* $\mathsf{v}_j, \mathsf{w}_j :: [\mathsf{p} \to \mathsf{q}_j]$, $1 \leq j \leq m$, *then* $\boldsymbol{\sigma}(\mathsf{v}_1 \sim \mathsf{w}_1, \ldots, \mathsf{v}_m \sim \mathsf{w}_m)$ *is well-formed.*

An element v *is* well-formed *if* $\mathsf{v} :: \mathsf{s}$ *for some* s. *The sets of well-formed types and well-formed elements are denoted by* $T(A, B, t, \equiv)$ *and* $E(A, B, t, \equiv)$, *respectively. We let* $TE(A, B, t, \equiv) = T(A, B, t, \equiv) \cup E(A, B, t, \equiv)$.

It follows from Definition 2 that if $\mathsf{v} :: \mathsf{s}$, then s is well-formed. Moreover, if $[\mathsf{p} \to \mathsf{q}] \equiv \mathsf{s}$ implies that $\mathsf{s} = [\mathsf{p}' \to \mathsf{q}']$, then we have that $\mathsf{v} :: \mathsf{s}$ implies that $\mathsf{s} = [\mathsf{p} \to \mathsf{q}]$ for some well-formed types p and q. Note that all elementary types are well-formed, i.e., we have $T(A) \subseteq T(A, B, t, \equiv)$. Also note that if $\equiv_1 \subseteq \equiv_2$, then $T(A, B, t, \equiv_1) \subseteq T(A, B, t, \equiv_2)$ and $E(A, B, t, \equiv_1) \subseteq E(A, B, t, \equiv_2)$. Moreover, we have the following properties, which we will need in the next section.

Proposition 1. *For* $k \geq 0$, *let* $\equiv_k$ *be equivalence relations with* $\equiv_k \subseteq \equiv_{k+1}$. *Then*

$$T(A, B, t, \bigcup_{k \geq 0} \equiv_k) = \bigcup_{k \geq 0} T(A, B, t, \equiv_k)$$

and

$$E(A, B, t, \bigcup_{k \geq 0} \equiv_k) = \bigcup_{k \geq 0} E(A, B, t, \equiv_k).$$

Proof. We only show the second; the first is analogous. To see that

$$\bigcup_{k \geq 0} E(A, B, t, \equiv_k) \subseteq E(A, B, t, \bigcup_{k \geq 0} \equiv_k)$$

let $\mathsf{v} \in \bigcup_{k \geq 0} E(A, B, t, \equiv_k)$. Then $\mathsf{v} \in E(A, B, t, \equiv_k)$ for some $k \geq 0$. Hence $\mathsf{v} \in E(A, B, t, \bigcup_{k \geq 0} \equiv_k)$ since $\equiv_k \subseteq \bigcup_{k \geq 0} \equiv_k$. To show the reverse, let $\equiv$ denote $\bigcup_{k \geq 0} \equiv_k$. Let $\mathsf{v} \in E(A, B, t, \equiv)$, i.e., let $\mathsf{v} :: \mathsf{s}$. To derive $\mathsf{v} :: \mathsf{s}$ from the cases (i) — (x), it is clear that (ix) and (x) cannot be used an infinite number of times, i.e., there is a finite list of equations $\mathsf{w}_i \equiv \mathsf{v}_i$ and a finite list of equations $\mathsf{s}_j \equiv \mathsf{s}'_j$ from which we can conclude that $\mathsf{v} :: \mathsf{s}$. But then, since $\equiv_k \subseteq \equiv_{k+1}$, there is a k such that $\mathsf{w}_i \equiv_k \mathsf{v}_i$ and $\mathsf{s}_j \equiv_k \mathsf{s}'_j$. That means that $\mathsf{v} \in E(A, B, t, \equiv_k) \subseteq \bigcup_{k \geq 0} E(A, B, t, \equiv_k)$. □

We stress that Definition 2 can be rewritten into the more constructive form of Definition 1 by indexing both the sets of well-formed types T_k as well as the typing relation; the latter by letting $:: = \bigcup_{k\geq 0} ::_k$, $::_0 = \emptyset$, and $::_k$ be defined cf. conditions (i) — (x). We think, however, that Definition 2 is more readable as it is.

The intuitive meaning of a well-formed term is that all of the operators it consists of are correctly applied with respect to the typing relation, in particular composition, product and the formation of subtypes and inclusions. Note that every type $\mathsf{a} \in A$ is well-formed, that $\mathsf{0}$ and $\mathsf{1}$ are well-formed, that an element $\mathsf{b} \in B$ is well-formed if $\mathsf{b} :: [\mathsf{p} \to \mathsf{q}]$ with well-formed p and elementary type q, and that $\mathsf{0}(\mathsf{p})$, $\mathsf{1}(\mathsf{p})$ and $\boldsymbol{\pi}_i(\mathsf{p}_1, \ldots, \mathsf{p}_n)$ are well-formed whenever p and p_i are well-formed, respectively.

Finally, note that Definition 2 excludes type assignments such as $\mathsf{u} :: [\boldsymbol{\sigma}(\mathsf{u}{\sim}\mathsf{u}) \to \mathsf{p}]$, at least if $[\boldsymbol{\sigma}(\mathsf{u}{\sim}\mathsf{u}) \to \mathsf{p}]$ is the only type that is associated with u. In general, type assignments such as $\mathsf{1}(\mathsf{0}) :: [\boldsymbol{\sigma}(\mathsf{1}(\mathsf{0}){\sim}\mathsf{1}(\mathsf{0})) \to \mathsf{1}]$ might be allowed though, viz. through condition (ix), provided that $[\mathsf{0} \to \mathsf{1}] \equiv [\boldsymbol{\sigma}(\mathsf{1}(\mathsf{0}){\sim}\mathsf{1}(\mathsf{0})) \to \mathsf{1}]$, since we already have $\mathsf{1}(\mathsf{0}) :: [\mathsf{0} \to \mathsf{1}]$ by condition (ii).

To understand the dynamics of Definition 2, in particular with respect to the given equivalence relation, consider the following example.

Example 1. Let $A = \{\mathsf{a}, \mathsf{a}'\}$ and let $B = \{\mathsf{b}\}$. Let $t(\mathsf{b}) = [\mathsf{a} \to \mathsf{a}']$. Consider the lists of types and elements below.

$\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b})$	$\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b})$
$\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}))$	$\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}))$
$\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b})))$	$\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b})))$
$\ldots$	$\ldots$

Note that every type is a member of $T(A, B)$ and every element is a member of $E(A, B)$. Also note that every element in the right column is a subterm of the type one row lower in the left column. To help the intuition of the reader, part of the situation is depicted below.

$$\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}) \xrightarrow[\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b})]{} \mathsf{a} \xrightarrow[\mathsf{b}]{} \mathsf{a}'$$

Now let $\equiv_I$ be the identity on $TE(A, B)$. According to conditions (xi) and (xii) of Definition 2 respectively, a, a' and $[\mathsf{a} \to \mathsf{a}']$ are well-formed, and so $\mathsf{b} :: [\mathsf{a} \to \mathsf{a}']$ by condition (i). Hence by condition (xv), $\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b})$ is well-formed, and $\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}) :: [\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}) \to \mathsf{a}]$ by condition (viii). Thus we have that $\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}) :: [\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}) \to \mathsf{a}']$ by condition (iii). However, the prerequisites of (xv) and (viii) respectively do not apply to the type $\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}))$ and the element $\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}\boldsymbol{\iota}(\mathsf{b}{\sim}\mathsf{b}))$ in the second row of the table above: note that it requires that the types $[\mathsf{a} \to \mathsf{a}']$ and $[\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}) \to \mathsf{a}']$ be equivalent. However, conditions (ix) and (x) are impotent in this respect. This means that only the first row above constitutes of a well-formed type (i.e., a member of $T(A, B, t, \equiv_I)$) and a well-formed element (a member of $E(A, B, t, \equiv_I)$). Note

that if we let $\equiv = \equiv_I \cup \{(\boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}), \mathsf{a}), (\mathsf{a}, \boldsymbol{\sigma}(\mathsf{b}{\sim}\mathsf{b}))\}$, then the types and elements in both the first and the second row are well-formed with respect to $T(A, B, t, \equiv)$ and $E(A, B, t, \equiv)$ respectively, but the type and element in the third row are not.

We need to further restrict well-formed elements and types, in particular because the dimensional structure $\boldsymbol{\delta}(\mathsf{w})$ induced by w is meaningful only if w can be interpreted as a inverse-finite function (see [8], or the Preliminaries). Also, for a class parameter $\boldsymbol{\gamma}(\mathsf{v})$, both $\boldsymbol{\gamma}(\mathsf{v})$ and v should be interpreted as functions that have a monoid as a codomain (again, see [8], or the Preliminaries). Well-formed elements and types thus formed will be called well-defined. We give the definitions.

Definition 3. *Let A and B be a set of basic type symbols and a set of basic element symbols respectively. Let t and $\equiv$ be a mapping and an equivalence relation respectively, cf. Definition 2. Let M be the set* $\{\mathsf{mon}, \mathsf{invfin}, \mathsf{inj}, \mathsf{hom}\}$ *of the* modalities *named* monoid*,* inverse-finite*,* injection *and* homomorphism*, respectively. Let* $u \subseteq (A \times \{\mathsf{mon}\}) \cup (B \times \{\mathsf{invfin}, \mathsf{inj}, \mathsf{hom}\})$ *be a relation. The* modality denomination induced by u*, denoted by* $\triangleright_u \subseteq TE(A, B) \times M$*, is defined as the smallest relation such that the following conditions hold:*

(i) *if* $(\mathsf{a}, \mathsf{mon}) \in u$*,* $\mathsf{a} \in A$*, then* $\mathsf{a} \triangleright \mathsf{mon}$*,*

(ii) *if* $(\mathsf{b}, \mathsf{m}) \in u$*,* $\mathsf{b} \in B$*, then* $\mathsf{b} \triangleright \mathsf{m}$*,*

(iii) $\mathsf{0}(\mathsf{p}) \triangleright \mathsf{inj}$ *and* $\mathsf{id}(\mathsf{p}) \triangleright \mathsf{inj}$*,*

(iv) *if* $\mathsf{v} \triangleright \mathsf{inj}$*, then* $\mathsf{v} \triangleright \mathsf{invfin}$*,*

(v) *if* $\mathsf{p}_i \triangleright \mathsf{mon}$*,* $1 \leq i \leq n$*, then* $\mathsf{p}_1 \times \cdots \times \mathsf{p}_n \triangleright \mathsf{mon}$*,*

(vi) *if* $\mathsf{v}, \mathsf{w} \triangleright \mathsf{m}$*, then* $\mathsf{v} \circ \mathsf{w} \triangleright \mathsf{m}$*,*

(vii) *if* $\mathsf{u}_i \triangleright \mathsf{m}$*,* $1 \leq i \leq n$*, then* $\langle \mathsf{u}_1, \ldots, \mathsf{u}_n \rangle \triangleright \mathsf{m}$*,*

(viii) $\boldsymbol{\iota}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_m{\sim}\mathsf{w}_m) \triangleright \mathsf{inj}$*,*

(ix) *if* $\mathsf{u} :: [1 \to \mathsf{p}]$ *then* $\mathsf{u} \triangleright \mathsf{inj}$*, and*

(x) *if* $\mathsf{y} \triangleright \mathsf{m}$ *and* $\mathsf{y} \equiv \mathsf{y}'$*, then* $\mathsf{y}' \triangleright \mathsf{m}$*,*

where in (i), $\mathsf{a} \triangleright \mathsf{mon}$ *is shorthand for* $(\mathsf{a}, \mathsf{mon}) \in \triangleright_{t,\equiv,u}$ *and similarly for (ii) — (x); where in (ii), (vi) and (vii),* $\mathsf{m} \in \{\mathsf{invfin}, \mathsf{inj}, \mathsf{hom}\}$*; and where in (x),* $\mathsf{m} \in \{\mathsf{mon}, \mathsf{invfin}, \mathsf{inj}, \mathsf{hom}\}$*.*

A type p *if* well-defined*, if* p *is well-formed and each proper subterm of* p *is well-defined. An element* u *is well-defined, if* u *is well-formed, each proper subterm of* u *is well-defined, and the following conditions hold:*

(xi) *if* $\mathsf{u} \triangleright \mathsf{hom}$ *and* $\mathsf{u} :: [\mathsf{p} \to \mathsf{q}]$*, then* $\mathsf{p} \triangleright \mathsf{mon}$ *and* $\mathsf{q} \triangleright \mathsf{mon}$*,*

(xii) *if* $\mathsf{u} = \boldsymbol{\delta}(\mathsf{w})$*, then* $\mathsf{w} \triangleright \mathsf{invfin}$*, and*

(xiii) *if* $\mathsf{u} = \boldsymbol{\gamma}(\mathsf{v})$ *and* $\mathsf{v} :: [\mathsf{p} \to \mathsf{q}]$*, then* $\mathsf{q} \triangleright \mathsf{mon}$*.*

The set of well-defined types is denoted $T(A, B, t, \equiv, u)$ *and the set of well-defined elements is denoted* $E(A, B, t, \equiv, u)$*; their union is* $TE(A, B, t, \equiv, u)$*.*

Definition 3 syntactically embodies some knowledge we have about monoids, inverse-finite functions, injections and homomorphisms, respectively. For instance, condition (iv) expresses that every injection is inverse-finite, and condition (v) expresses that monoids are closed under products of types. Conditions (vi) and (vii) express that injections, inverse-finite functions and homomorphisms are closed under compositions and products, respectively.

Note that if $\boldsymbol{\gamma}(\mathsf{v})$ is well-defined, then we can, e.g., conclude that $\boldsymbol{\gamma}(\mathsf{v}) :: [\mathsf{F}(\mathsf{p}) \to \mathsf{q}]$ with $\mathsf{q} \triangleright \mathsf{mon}$, as desired.

It is possible to extend the modalities of Definition 3 with notions that may have other significance to statistics, or to statistical metadata in particular. For instance, we could introduce modalities corresponding to the notion of an *object type* [8] and an *object type relation* [8], and add the condition to Definition 3 that an object type relation v (i.e., any element v that is denominated as an object type relation) with type $[\mathsf{p} \to \mathsf{q}]$ is well-defined if both p and q are denominated as an object type. Also, given a meaningful denomination corresponding to the notion of a statistical variable, we could add the condition to Definition 3 that a type $\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{w})$ is an object type if v and w are variables. Though these additional notions could prove useful (and we have introduced the mechanisms for formalizing them in Definition 3) we stress that they play no significant role in the theory developed in this article. That changes if we added the notion of a classification system as a type modality, since we then would have to add the rules of a Boolean algebra (see [7, 8]) to our notion of a congruence of elements and types (to be defined later in the article).

Also, it may seem odd that we rather superficially introduced the notion of a monoid, since we left out the corresponding monoid operation and unit. Again, these could be added to our grammar of elements, and then we could add the monoid laws to our notion of congruence (defined later). However, as far as structural metadata is concerned, that would become significant once we also added the notion of *sum* (called *row-wise combination* in [8]) because only the congruence laws involving sum need the monoid operation. We stress again however that the mechanisms to formally add the notion of a monoid have been introduced here (or will be introduced shortly), but we decided to leave them them out of the theory for the sake of simplicity and brevity.

With respect to well-defined types and elements, we have a corollary similar to Proposition 1.

Corollary 1. *Let, for $k \geq 0$, $\equiv_k$ be equivalence relations with $\equiv_k \subseteq \equiv_{k+1}$. Then*

$$T(A, B, t, \bigcup_{k\geq 0} \equiv_k, u) = \bigcup_{k\geq 0} T(A, B, t, \equiv_k, u)$$

and

$$E(A, B, t, \bigcup_{k\geq 0} \equiv_k, u) = \bigcup_{k\geq 0} E(A, B, t, \equiv_k, u).$$

Proof. Analogous to the proof of Proposition 1. □

The last notion that we define in this section is the general notion of a congruence on well-defined types and elements, given a typing relation and a denomination. Intuitively, a congruence relates terms that should be considered equal; in the next section we give the (business) rules that motivate on what grounds terms are equal and we define one particular congruence relation with them.

Definition 4. *Given sets of basic type and basic element symbols A and B; a mapping and an equivalence relation t and $\equiv$ respectively, cf. Definition 2; and a relation u cf. Definition 3; a relation $\cong \subseteq T(A,B,t,\equiv,u)^2 \cup E(A,B,t,\equiv,u)^2$ is a* congruence*, if $\cong$ is an equivalence relation that is closed by the constructs of Definition 1, i.e., for all well-defined terms x, y and z we have*

(i) $\mathsf{x} \cong \mathsf{x}$,

(ii) if $\mathsf{x} \cong \mathsf{y}$, *then* $\mathsf{y} \cong \mathsf{x}$,

(iii) if $\mathsf{x} \cong \mathsf{y}$ *and* $\mathsf{y} \cong \mathsf{z}$, *then* $\mathsf{x} \cong \mathsf{z}$,

and we have

(iv) if $\mathsf{p} \cong \mathsf{p}'$ *and* $\mathsf{q} \cong \mathsf{q}'$, *then* $[\mathsf{p} \to \mathsf{q}] \cong [\mathsf{p}' \to \mathsf{q}']$,

(v) if $\mathsf{p}_i \cong \mathsf{p}'_i$ *for all* $1 \leq i \leq n$, *then* $\mathsf{p}_1 \times \cdots \times \mathsf{p}_n \cong \mathsf{p}'_1 \times \cdots \times \mathsf{p}'_n$,

(vi) if $\mathsf{p} \cong \mathsf{p}'$, *then* $\mathsf{F}(\mathsf{p}) \cong \mathsf{F}(\mathsf{p}')$,

(vii) if $\mathsf{v}_j \cong \mathsf{v}'_j$ *and* $\mathsf{w}_j \cong \mathsf{w}'_j$ *for all* $1 \leq j \leq m$, *then*
$\boldsymbol{\sigma}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_m{\sim}\mathsf{w}_m) \cong \boldsymbol{\sigma}(\mathsf{v}'_1{\sim}\mathsf{w}'_1, \ldots, \mathsf{v}'_m{\sim}\mathsf{w}'_m)$,

(viii) if $\mathsf{p} \cong \mathsf{p}'$, *then* $\mathsf{0}(\mathsf{p}) \cong \mathsf{0}(\mathsf{p}')$, $\mathsf{1}(\mathsf{p}) \cong \mathsf{1}(\mathsf{p}')$ *and* $\mathsf{id}(\mathsf{p}) \cong \mathsf{id}(\mathsf{p}')$,

(ix) if $\mathsf{v} \cong \mathsf{v}'$ *and* $\mathsf{w} \cong \mathsf{w}'$, *then* $\mathsf{v} \circ \mathsf{w} \cong \mathsf{v}' \circ \mathsf{w}'$,

(x) if $\mathsf{u}_i \cong \mathsf{u}'_i$ *for all* $1 \leq i \leq n$, *then* $\langle \mathsf{u}_1, \ldots, \mathsf{u}_n \rangle \cong \langle \mathsf{u}'_1, \ldots, \mathsf{u}'_n \rangle$,

(xi) if $\mathsf{p}_i \cong \mathsf{p}'_i$ *for all* $1 \leq i \leq n$, *then* $\boldsymbol{\pi}_i(\mathsf{p}_1, \ldots, \mathsf{p}_n) \cong \boldsymbol{\pi}_i(\mathsf{p}'_1, \ldots, \mathsf{p}'_n)$,

(xii) if $\mathsf{w} \cong \mathsf{w}'$, *then* $\boldsymbol{\delta}(\mathsf{w}) \cong \boldsymbol{\delta}(\mathsf{w}')$,

(xiii) if $\mathsf{v} \cong \mathsf{v}'$, *then* $\boldsymbol{\gamma}(\mathsf{v}) \cong \boldsymbol{\gamma}(\mathsf{v}')$, *and*

(xiv) if $\mathsf{v}_j \cong \mathsf{v}'_j$ *and* $\mathsf{w}_j \cong \mathsf{w}'_j$ *for all* $1 \leq j \leq m$, *then*
$\boldsymbol{\iota}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_m{\sim}\mathsf{w}_m) \cong \boldsymbol{\iota}(\mathsf{v}'_1{\sim}\mathsf{w}'_1, \ldots, \mathsf{v}'_m{\sim}\mathsf{w}'_m)$,

where it is understood that the left and the right hand sides of equations (iv) – (xiv) yield well-defined terms. We say that $\cong$ is a congruence on $T(A,B,t,\equiv,u)^2 \cup E(A,B,t,\equiv,u)^2$.

Note that we thus require that the left and the right hand sides of two terms that are congruent are well-defined. Also note that we don't require that the left and the right hand sides of a congruence, in the case both are elements, have a common type according to the typing relation $::_{t,\equiv}$. In fact, the congruences we will establish in the next section will not have this property in general. Instead, they will satisfy the following: if $\mathsf{v} \cong \mathsf{w}$, $\mathsf{v} :: \mathsf{s}$ and $\mathsf{w} :: \mathsf{s}'$, then $\mathsf{s} \cong \mathsf{s}'$.

In the following section, Definition 4 is used to construct a particular congruence, equating pairs of terms that are presumed equivalent, cf. the equations of Sections 1, 2 and 3. For instance, we want to include into the congruence we have in mind, all well-defined terms of the form $\mathsf{idv} \cong \mathsf{v}$ and $\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{v}) \cong \mathsf{id}$.

5 A formal language for structural metadata

In this section we incorporate into our language the rules discovered in Sections 2 and 3. We define a family of congruences, based on a family of notions of well-definedness, and give a closure property to define a final congruence and a final notion of well-definedness. We show that these final notions satisfy natural and desired properties.

Definition 5. *Let A, B, t, $\equiv$, and u be given, cf. Definition 4. The relation* $\mathrm{Con}(t, \equiv, u) \subseteq T(A, B, t, \equiv, u)^2 \cup E(A, B, t, \equiv, u)^2$*, called the* congruence generated by t, $\equiv$ and u*, is defined as the smallest congruence $\cong$ that contains the pairs below:*

$$\mathsf{p}_1 \times \cdots \times \mathsf{0} \times \cdots \times \mathsf{p}_n \cong \mathsf{0} \tag{0a}$$
$$\mathsf{v} \cong \mathsf{0}(\mathsf{q}),\ \textit{provided}\ \mathsf{v} :: [\mathsf{0} \to \mathsf{q}] \tag{0b}$$
$$\mathsf{v} \cong \mathsf{1}(\mathsf{p}),\ \textit{provided}\ \mathsf{v} :: [\mathsf{p} \to \mathsf{1}] \tag{0c}$$
$$\mathsf{idv} \cong \mathsf{v} \tag{0d}$$
$$\mathsf{vid} \cong \mathsf{v} \tag{0e}$$

$$\mathsf{u}(\mathsf{vw}) \cong (\mathsf{uv})\mathsf{w} \tag{1}$$
$$\boldsymbol{\pi}_i\langle \mathsf{v}_1, \ldots, \mathsf{v}_n\rangle \cong \mathsf{v}_i \tag{2'}$$
$$\langle \mathsf{v}_1\mathsf{w}, \ldots, \mathsf{v}_n\mathsf{w}\rangle \cong \langle \mathsf{v}_1, \ldots, \mathsf{v}_n\rangle\mathsf{w} \tag{3'}$$
$$\boldsymbol{\alpha}(\boldsymbol{\alpha}(\mathsf{v}, \mathsf{w}), \mathsf{u}) \cong \boldsymbol{\alpha}(\mathsf{v}, \mathsf{uw}) \tag{4}$$
$$\mathsf{u}\boldsymbol{\alpha}(\mathsf{v}, \mathsf{w}) \cong \boldsymbol{\alpha}(\mathsf{uv}, \mathsf{w}),\ \textit{provided}\ \mathsf{u} \triangleright \mathsf{hom} \tag{5}$$
$$\langle \boldsymbol{\alpha}(\mathsf{v}_1, \mathsf{w}), \ldots, \boldsymbol{\alpha}(\mathsf{v}_n, \mathsf{w})\rangle \cong \boldsymbol{\alpha}(\langle \mathsf{v}_1, \ldots, \mathsf{v}_n\rangle, \mathsf{w}) \tag{6'}$$
$$\boldsymbol{\alpha}(\mathsf{v}, \mathsf{w})\mathsf{w} \cong \mathsf{v},\ \textit{provided}\ \mathsf{w} \triangleright \mathsf{inj} \tag{7}$$

$$\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}) \cong \mathsf{w}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}) \tag{8}$$
$$\boldsymbol{\sigma}(\mathsf{uv}, \mathsf{uw}) \cong \boldsymbol{\sigma}(\mathsf{v}, \mathsf{w}),\ \textit{provided}\ \mathsf{u} \triangleright \mathsf{inj} \tag{11}$$
$$\boldsymbol{\iota}(\mathsf{uv}{\sim}\mathsf{uw}) \cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}),\ \textit{provided}\ \mathsf{u} \triangleright \mathsf{inj} \tag{12}$$
$$\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{w}) \cong \boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{w}) \tag{13}$$
$$\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{w}) \cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}) \tag{14}$$
$$\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{d1}, \mathsf{v}{\sim}\mathsf{e1}) \cong \mathsf{0},\ \textit{provided}\ \mathsf{d}, \mathsf{e} \in B\ \textit{with}\ \mathsf{d} \neq \mathsf{e} \tag{15}$$
$$\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{d1}, \mathsf{v}{\sim}\mathsf{e1}) \cong \mathsf{0},\ \textit{provided}\ \mathsf{d}, \mathsf{e} \in B\ \textit{with}\ \mathsf{d} \neq \mathsf{e} \tag{16}$$
$$\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{v}) \cong \mathsf{p},\ \textit{provided}\ \mathsf{v} :: [\mathsf{p} \to \mathsf{q}] \tag{17}$$
$$\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{v}) \cong \mathsf{id}(\mathsf{p}),\ \textit{provided}\ \mathsf{v} :: [\mathsf{p} \to \mathsf{q}] \tag{18}$$
$$\boldsymbol{\sigma}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_m{\sim}\mathsf{w}_m) \cong \boldsymbol{\sigma}(\langle \mathsf{v}_1, \ldots, \mathsf{v}_m\rangle{\sim}\langle \mathsf{w}_1, \ldots, \mathsf{w}_m\rangle) \tag{23}$$
$$\boldsymbol{\iota}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_m{\sim}\mathsf{w}_m) \cong \boldsymbol{\iota}(\langle \mathsf{v}_1, \ldots, \mathsf{v}_m\rangle{\sim}\langle \mathsf{w}_1, \ldots, \mathsf{w}_m\rangle) \tag{24}$$
$$\boldsymbol{\sigma}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_{m+1}{\sim}\mathsf{w}_{m+1}) \cong \boldsymbol{\sigma}(\mathsf{v}_{m+1}\boldsymbol{\iota}_1^m{\sim}\mathsf{w}_{m+1}\boldsymbol{\iota}_1^m) \tag{25}$$
$$\boldsymbol{\iota}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_{m+1}{\sim}\mathsf{w}_{m+1}) \cong \boldsymbol{\iota}_1^m\boldsymbol{\iota}(\mathsf{v}_{m+1}\boldsymbol{\iota}_1^m{\sim}\mathsf{w}_{m+1}\boldsymbol{\iota}_1^m) \tag{26}$$
$$\boldsymbol{\sigma}(\mathsf{v}_1{\sim}\mathsf{w}_1, \ldots, \mathsf{v}_m{\sim}\mathsf{w}_m) \cong \boldsymbol{\sigma}(\mathsf{v}_{\phi(1)}{\sim}\mathsf{w}_{\phi(1)}, \ldots, \mathsf{v}_{\phi(m)}{\sim}\mathsf{w}_{\phi(m)}) \tag{27}$$

$$\boldsymbol{\iota}(\mathsf{v}_1{\sim}\mathsf{w}_1,\ldots,\mathsf{v}_m{\sim}\mathsf{w}_m) \cong \boldsymbol{\iota}(\mathsf{v}_{\phi(1)}{\sim}\mathsf{w}_{\phi(1)},\ldots,\mathsf{v}_{\phi(m)}{\sim}\mathsf{w}_{\phi(m)}) \tag{28}$$

$$\boldsymbol{\alpha}(\mathsf{v},\mathsf{w})\boldsymbol{\iota}(\mathsf{u}{\sim}\mathsf{z}) \cong \boldsymbol{\alpha}(\mathsf{v}\boldsymbol{\iota}(\mathsf{uw}{\sim}\mathsf{zw}),\mathsf{w}\boldsymbol{\iota}(\mathsf{uw}{\sim}\mathsf{zw}))\boldsymbol{\iota}(\mathsf{u}{\sim}\mathsf{z}) \tag{29}$$

$$\boldsymbol{\alpha}(\mathsf{v},\mathsf{w}) \circ \mathsf{d} \cong \boldsymbol{\alpha}(\mathsf{v}\boldsymbol{\iota}(w{\sim}\mathsf{d1}),\mathsf{w}\boldsymbol{\iota}(\mathsf{w}{\sim}\mathsf{d1})) \circ \mathsf{d}, \tag{30}$$

where ϕ is any permutation of $\{1,\ldots,m\}$ and $\boldsymbol{\iota}(\mathsf{v}_1{\sim}\mathsf{w}_1,\ldots,\mathsf{v}_m{\sim}\mathsf{w}_m)$ is abbreviated by $\boldsymbol{\iota}_1^m$.

When t, $\equiv$ and u are clear from the context, we let $\mathsf{v} \cong \mathsf{w}$ be an abbreviation of $(\mathsf{v},\mathsf{w}) \in \mathrm{Con}(t,\equiv,u)$.

In Definition 5, the intention of laws (0a) — (0e) should be clear from the Preliminaries. Observe that laws (0b) and (0c) comprise many different situations: $\mathsf{0}(\mathsf{1}) \cong \mathsf{1}(\mathsf{0})$, $\mathsf{id}(\mathsf{1}) \cong \mathsf{1}(\mathsf{1})$ and $\mathsf{1}(\mathsf{q})\mathsf{w} \cong \mathsf{1}(\mathsf{p})$, provided $\mathsf{w} :: [\mathsf{p} \to \mathsf{q}]$, are some instances of them. Note that laws (0d) and (0e) correspond to the left and right identities for composition, as mentioned in the Preliminaries. Laws (1) — (7) correspond to Equations 1 to 7 at the end of the Preliminaries. Laws (8) — (30) correspond to equations with identical numbers from Sections 2 and 3.

Note that, by Definition 4, for Definition 5 to make sense, it is required that both the left and the right hand side of each equation yield well-defined terms. Also note that, for reasons of brevity, shorthand notation is used in some of the laws, leaving out, e.g., arguments of the id, $\mathsf{0}$ and $\boldsymbol{\pi}_i$ elements, as for instance in laws (0d), (0e), (16) and law (2'). Finally note that each equation that involves elements either has identical types on the left and the right hand side, or it can be deduced from $\cong$ that they have identical types. So, for instance, in law (16), the left hand side has type $[\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{d1},\mathsf{v}{\sim}\mathsf{e1}) \to \mathsf{p}]$ (provided, e.g., $\mathsf{v} :: [\mathsf{p} \to \mathsf{q}]$) which reduces to $[\mathsf{0} \to \mathsf{p}]$ by law (15).

We stress that many of the laws of Definition 5 are families of laws, for instance law (0a) for each n and for each position of $\mathsf{0}$ in the left hand side, and law (2') for each n and i, and for each $\boldsymbol{\pi}_i(\mathsf{p}_1,\ldots,\mathsf{p}_n)$, i.e., for each combination of $\mathsf{p}_1,\ldots,\mathsf{p}_n$, and for each suitable combination of $\mathsf{v}_1,\ldots,\mathsf{v}_n$. Note that there is no need to extend law 29 to a family of laws, i.e., one that contains $\boldsymbol{\iota}(\mathsf{u}_1{\sim}\mathsf{z}_1,\ldots,\mathsf{u}_m{\sim}\mathsf{z}_m)$ and $\boldsymbol{\iota}(\mathsf{u}_1\mathsf{w}{\sim}\mathsf{z}_1\mathsf{w},\ldots,\mathsf{u}_m\mathsf{w}{\sim}\mathsf{z}_m\mathsf{w})$ instead of $\boldsymbol{\iota}(\mathsf{u}{\sim}\mathsf{z})$ and $\boldsymbol{\iota}(\mathsf{uw}{\sim}\mathsf{zw})$, because of laws (24) and (3').

Observe that laws (25) and (26) are formulated in a slightly different, but equivalent, way compared to the versions formulated in Section 3. Also observe that law (14) is formulated differently, viz. using law (26) applied to $\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w},\mathsf{v}{\sim}\mathsf{w})$, together with (0e).

Note the conditions of laws (15) and (16): by $\mathsf{d} \neq \mathsf{e}$ we mean that d and e are unequal as terms, i.e., d and e are different basic element symbols. It might be tempting to extend these conditions by dropping the requirement $\mathsf{d},\mathsf{e} \in B$, i.e., by requiring that d and e be any suitable well-defined terms.[2] This would yield a problematic semantics though: take for instance $\mathsf{d} = \mathsf{vd}'$ and $\mathsf{e} = \mathsf{ve}'$ with $\mathsf{d}',\mathsf{e}' \in B$ with $\mathsf{d}' \neq \mathsf{e}'$. Now even if we adopt as a requirement that different basic element symbols of type $[\mathsf{1} \to \mathsf{p}]$ represent different values from the set associated with p

[2] One might be equally tempted to require that $\mathsf{d} \ncong \mathsf{e}$, instead of $\mathsf{d} \neq \mathsf{e}$. This would be technically challenging, and equally wrong.

(which we will do in Section 6), then we cannot conclude that vd' and ve' represent different values, in much the same way that from $x \neq y$ we cannot conclude that $f(x) \neq f(y)$.

Finally, we deduce that laws (13), (14), (17) and (18), in combination with laws (27) and (28), and laws (0d) and (0e), express that arguments of $\boldsymbol{\iota}$ and $\boldsymbol{\sigma}$ form a set of pairs $\mathsf{v}{\sim}\mathsf{w}$ from which pairs of the form $\mathsf{v}{\sim}\mathsf{v}$ can be excluded. To see that, we consider the following expansion:

$$\begin{aligned}
\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{v}, \mathsf{v}{\sim}\mathsf{w}) &\cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{v})\boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{v}){\sim}\mathsf{w}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{v})) \\
&\cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}), \mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})) \circ \\
&\quad \boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}), \mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})), \\
&\quad\quad \mathsf{w}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}), \mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}))) \\
&\cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\mathsf{id}\boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\mathsf{id}, \mathsf{w}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\mathsf{id}) \\
&\cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})\boldsymbol{\iota}(\mathsf{v}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}), \mathsf{w}\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w})) \\
&\cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}, \mathsf{v}{\sim}\mathsf{w}) \\
&\cong \boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{w}),
\end{aligned}$$

where we used, respectively, laws (26) from left to right twice, law (18), law (0e), law (26) from right to left, and finally law (14).

Note that we have that $\equiv_1 \subseteq \equiv_2$ implies that $\mathrm{Con}(t, \equiv_1, u) \subseteq \mathrm{Con}(t, \equiv_2, u)$. This is used in Proposition 2 below.

Next we define the closure of a family of congruences that is built up using Definitions 2 and 3, together with Definition 5.

Definition 6. *Let A, B, t, $\equiv$, and u be given, cf. Definition 5. Let $\equiv_I$ be the identity on $TE(A,B)$. The relation $\cong \subseteq T(A,B)^2 \cup E(A,B)^2$ is defined as*

$$\cong = \bigcup_{k \geq 0} \cong_k$$

with $\cong_0 = \emptyset$, and

$$\cong_k = \mathrm{Con}(t, \equiv_I \cup \cong_{k-1}, u)$$

for $k > 0$.

The closure construction of Definition 6 gives us the final notions of congruence and of well-definedness. This is stated below.

Proposition 2. *Let $\cong_k$ and $\cong$ be as in Definition 6. Then we have*

(i) $\cong_k \subseteq \cong_{k+1}$ for all $k \geq 0$, and

(ii) $\cong = \mathrm{Con}(t, \equiv_I \cup \cong, u)$,

i.e., $\cong$ is the smallest congruence on $T(A,B,t,\equiv_I \cup \cong, u)^2 \cup E(A,B,t,\equiv_I \cup \cong, u)^2$ that satisfies the laws of Definition 5.

Proof. Property (i) is shown by induction on k, using the remark just above Definition 6. To prove inclusion of (ii) from right to left, it suffices to show that $\cong$ is a congruence on $T(A,B,t,\equiv_I \cup \cong, u)^2 \cup E(A,B,t,\equiv_I \cup \cong, u)^2$ that satisfies the laws of Definition 5. Since $\mathrm{Con}(t,\equiv_I \cup \cong, u)$ is the smallest such congruence, we have $\mathrm{Con}(t,\equiv_I \cup \cong, u) \subseteq \cong$. By Corollary 1 and (i) we have that $\cong$ is a relation on $T(A,B,t,\equiv_I \cup \cong, u)^2 \cup E(A,B,t,\equiv_I \cup \cong, u)^2$. To prove that $\cong$ is a congruence, we need to show properties (i) — (xiv) of Definition 4. All are proven similarly: to show (iii) for instance, let $\mathsf{x} \cong \mathsf{y}$ and $\mathsf{y} \cong \mathsf{z}$. Then $\mathsf{x} \cong_i \mathsf{y}$ and $\mathsf{y} \cong_j \mathsf{z}$ for some $i,j \geq 0$. Hence $\mathsf{x} \cong_k \mathsf{y}$ and $\mathsf{y} \cong_k \mathsf{z}$ with $k = \max(i,j)$. Since $\cong_k$ is a congruence, we have $\mathsf{x} \cong_k \mathsf{z}$ and hence $\mathsf{x} \cong \mathsf{z}$. To show that $\cong$ satisfies the laws of Definition 5, let $\mathsf{x},\mathsf{y} \in TE(A,B,t,\equiv_I \cup \cong, u)$ be a pair of types or elements that satisfy one of the laws. By Corollary 1 and (i), $\mathsf{x},\mathsf{y} \in TE(A,B,t,\equiv_I \cup \cong_k, u)$ for some k. Hence $\mathsf{x} \cong_{k+1} \mathsf{y}$ and hence $\mathsf{x} \cong \mathsf{y}$. The inclusion of (ii) in the converse direction is immediate by the remark just above Definition 6. □

The following properties show the soundness of the construction of Definition 6 and earlier definitions. They show that the typing relation behaves as expected: that elements are all assigned function types, that the typing relation of Definition 2 and the modality denomination of Definition 3 are closed by congruence in a natural way.

Proposition 3. *Let* $\mathsf{r},\mathsf{p},\mathsf{q},\mathsf{s},\mathsf{s}',\mathsf{x},\mathsf{y},\mathsf{v},\mathsf{w} \in TE(A,B,t,\equiv_I \cup \cong, u)$. *Then we have*

(i) If $\mathsf{r} \cong [\mathsf{p} \to \mathsf{q}]$, *then* $\mathsf{r} = [\mathsf{p}' \to \mathsf{q}']$ *with* $\mathsf{p} \cong \mathsf{p}'$ *and* $\mathsf{q} \cong \mathsf{q}'$,

(ii) If $\mathsf{v} :: \mathsf{s}$, *then* $\mathsf{s} = [\mathsf{p} \to \mathsf{q}]$ *for some* p *and* q,

(iii) If $\mathsf{v} :: \mathsf{s}$, *then* $\mathsf{v} :: \mathsf{s}'$ *if and only if* $\mathsf{s} \cong \mathsf{s}'$,

(iv) If $\mathsf{v} :: \mathsf{s}$ *and* $\mathsf{w} \cong \mathsf{v}$, *then* $\mathsf{w} :: \mathsf{s}$,

(v) If $\mathsf{y} \triangleright \mathsf{m}$ *and* $\mathsf{y} \cong \mathsf{y}'$, *then* $\mathsf{y}' \triangleright \mathsf{m}$.

Proof. The proofs of (iii) — (v) are immediate by Proposition 2(ii) and by substitution of $\equiv_I \cup \cong$ for $\equiv$ in Definition 2(ix) and (x), and in Definition 3(x), respectively. The proof of (ii) is immediate by (i) and the fact that in (i) — (viii) of Definition 2, a type assignment of the form $\mathsf{v} :: [\mathsf{p} \to \mathsf{q}]$ is concluded. To prove (i), note that the only laws from Definitions 4 and 5 from which $\mathsf{r} \cong [\mathsf{p} \to \mathsf{q}]$ can be concluded, are laws (i) — (iv) of Definition 4. Induction to the length of the derivation of $\mathsf{r} \cong [\mathsf{p} \to \mathsf{q}]$ shows property (i). □

6 A sketch of semantics

In this section we give a sketch of the sets-and-functions semantics of the language developed in the previous two sections, relating the material developed there with that of Sections 2 and 3.

We assume possibly empty, countable and disjoint sets A and B of basic type and basic element symbols, a mapping $t : B \to T(A,B)$ cf. Definition 2, and a relation $u \subseteq (A \times \{\mathsf{mon}\}) \cup (B \times \{\mathsf{invfin}, \mathsf{inj}, \mathsf{hom}\})$ cf. Definition 3.

Let, for every $\mathsf{a} \in A$, $\mathcal{I}(\mathsf{a})$ be a nonempty set, let $\mathcal{I}(\mathsf{0}) = \emptyset$, the empty set, and let $\mathcal{I}(\mathsf{1}) = \{*\}$, an arbitrary but fixed one-element set.[3] If $(\mathsf{a}, \mathsf{mon}) \in u$, then we assume that $\mathcal{I}(\mathsf{a})$ comes with a proper associative and commutative binary operation $+$ and with a proper identity 0, i.e., we then treat $\mathcal{I}(\mathsf{a})$ as a commutative monoid, cf. [8].

Let the set $\mathcal{D}_0$ be the collection of all $\mathcal{I}(\mathsf{a})$ together with $\mathcal{I}(\mathsf{0})$ and $\mathcal{I}(\mathsf{1})$, i.e., $\mathcal{D}_0 = \{\mathcal{I}(\mathsf{a}) \mid \mathsf{a} \in A\} \cup \{\emptyset\} \cup \{\{*\}\}$.[4] Let $\mathcal{D}'$ be the smallest set that contains $\mathcal{D}_0$ and that is closed by the formation of arbitrary Cartesian products, finite power sets, subsets, and set exponentiation, i.e., $\mathcal{D}_0 \subseteq \mathcal{D}'$, and

(i) if $x_1, \ldots, x_n \in \mathcal{D}'$, then $x_1 \times \cdots \times x_n \in \mathcal{D}'$,

(ii) if $x \in \mathcal{D}'$, then $Fx \in \mathcal{D}'$,

(iii) if $x \in \mathcal{D}'$ and $y \subseteq x$, then $y \in \mathcal{D}'$, and

(iv) if $x, y \in \mathcal{D}'$, then $x^y \in \mathcal{D}'$.

Finally, let $\mathcal{D}'' = \bigcup \mathcal{D}'$ and let $\mathcal{D} = \mathcal{D}' \cup \mathcal{D}''$.[5] We note that $\mathcal{I}$ is a mapping $A \cup \{\mathsf{0}, \mathsf{1}\} \to \mathcal{D}_0$. We extend $\mathcal{I}$ to a mapping $TE(A, B, t, \equiv_I \cup \cong, u) \to \mathcal{D}$ next.

Following the grammar of the informal definition of types and elements prior to Definition 1, we let $\mathcal{I}(\mathsf{p})$ and $\mathcal{I}(\mathsf{v})$, with p and v well-defined, be compositional in the way expected:

$$\begin{aligned}\mathcal{I}(\mathsf{p}) \quad ::= \quad & \mathcal{I}(\mathsf{a}) \mid \mathcal{I}(\mathsf{0}) \mid \mathcal{I}(\mathsf{1}) \mid \mathcal{I}(\mathsf{q})^{\mathcal{I}(\mathsf{p})} \mid \mathcal{I}(\mathsf{p}_1) \times \cdots \times \mathcal{I}(\mathsf{p}_n) \mid \\ & F\mathcal{I}(\mathsf{p}) \mid \sigma(\mathcal{I}(\mathsf{v}_1) \sim \mathcal{I}(\mathsf{w}_1), \ldots, \mathcal{I}(\mathsf{v}_m) \sim \mathcal{I}(\mathsf{w}_m))\end{aligned}$$

and

$$\begin{aligned}\mathcal{I}(\mathsf{v}) \quad ::= \quad & \mathcal{I}(\mathsf{b}) \mid \mathcal{I}(\mathsf{0}(\mathsf{p})) \mid \mathcal{I}(\mathsf{1}(\mathsf{p})) \mid \mathcal{I}(\mathsf{id}(\mathsf{p})) \mid \mathcal{I}(\mathsf{v}) \circ \mathcal{I}(\mathsf{w}) \mid \\ & \langle \mathcal{I}(\mathsf{u}_1), \ldots, \mathcal{I}(\mathsf{u}_n) \rangle \mid \pi_i^n(\mathcal{I}(\mathsf{p}_1), \ldots, \mathcal{I}(\mathsf{p}_n)) \mid \\ & \gamma(\mathcal{I}(\mathsf{v})) \mid \delta(\mathcal{I}(\mathsf{w})) \mid \iota(\mathcal{I}(\mathsf{v}_1) \sim \mathcal{I}(\mathsf{w}_1), \ldots, \mathcal{I}(\mathsf{v}_m) \sim \mathcal{I}(\mathsf{w}_m)),\end{aligned}$$

where $\mathcal{I}(\mathsf{0}(\mathsf{p}))$ is $0_{\mathcal{I}(\mathsf{p})}$, $\mathcal{I}(\mathsf{1}(\mathsf{p}))$ is $1_{\mathcal{I}(\mathsf{p})}$, $\mathcal{I}(\mathsf{id}(\mathsf{p}))$ is the identity on $\mathcal{I}(\mathsf{p})$, and $\mathcal{I}(\mathsf{b})$ is a function that respects the elementary typing mapping t and the relation u, i.e., $\mathcal{I}(\mathsf{b})$ is an element of $\mathcal{I}(\mathsf{q})^{\mathcal{I}(\mathsf{p})}$ if $t(\mathsf{b}) = [\mathsf{p} \to \mathsf{q}]$, that is inverse-finite, an injection or a homomorphism, whenever $(\mathsf{b}, \mathsf{invfin}) \in u$, $(\mathsf{b}, \mathsf{inj}) \in u$, or $(\mathsf{b}, \mathsf{hom}) \in u$, respectively. Note that we thus assume that $\mathcal{I}$, t and u 'work together' well; for instance, if $(\mathsf{b}, \mathsf{inj}) \in u$, then the cardinalities of $\mathcal{I}(\mathsf{p})$ and $\mathcal{I}(\mathsf{q})$ must be such that an injection $\mathcal{I}(\mathsf{p}) \to \mathcal{I}(\mathsf{q})$ indeed exists. Also, if b, b' are different elementary type symbols with $t(\mathsf{b}) = t(\mathsf{b}') = [\mathsf{1} \to \mathsf{q}]$, then we assume that $\mathcal{I}(\mathsf{b}) \neq \mathcal{I}(\mathsf{b}')$.

We claim that $\mathcal{I}$ is well-defined and has the expected properties.

Proposition 4. *The mapping $\mathcal{I}$ is well-defined and sound, i.e., we have*

(i) $\mathcal{I} : TE(A, B, t, \equiv_I \cup \cong, u) \to \mathcal{D}$,

[3] We use $\mathcal{I}$ to indicate *interpretation*.

[4] We use $\mathcal{D}$ to indicate *data*.

[5] $\mathcal{D}'' = \bigcup \mathcal{D}'$ means that $\mathcal{D}'' = \{d \mid d \in x \text{ and } x \in \mathcal{D}'\}$

and for all well-defined types p *and* q *and all well-defined elements* v *and* w*, we have*

(ii) if $\mathsf{v} :: [\mathsf{p} \to \mathsf{q}]$*, then* $\mathcal{I}(\mathsf{q})^{\mathcal{I}(\mathsf{p})}$ *is nonempty and contains* $\mathcal{I}(\mathsf{v})$*,*

(iii) if $\mathsf{p} \triangleright \mathsf{mon}$*, then* $\mathcal{I}(\mathsf{p})$ *is a commutative monoid,*

(iv) if $\mathsf{v} \triangleright \mathsf{invfin}$*, then* $\mathcal{I}(\mathsf{v})$ *is inverse-finite, and similarly for the cases* $\mathsf{v} \triangleright \mathsf{inj}$ *and* $\mathsf{v} \triangleright \mathsf{hom}$*, and*

(v) if $\mathsf{v} \cong \mathsf{w}$*, then* $\mathcal{I}(\mathsf{v}) = \mathcal{I}(\mathsf{w})$*, and if* $\mathsf{p} \cong \mathsf{q}$*, then* $\mathcal{I}(\mathsf{p}) = \mathcal{I}(\mathsf{q})$*.*

The proof of Proposition 4 is left to the reader.

7 Examples

In this section we give some examples that show the expressiveness of the language defined in the previous sections. We start with showing how to incorporate families of variables, indexed by a list of categories, into the language.

Example 2. In some statistics within a statistical office families of likewise variables are used, e.g., to record the answers to questions in a questionnaire. In the DSC there is no possibility to record the meaning of these variables in one stroke. Instead one must give definitions for each of the variables individually, even if these definitions show minimal differences. Thus the administrative burden is increased, as well as the risk of errors and inconsistencies. Especially in the case in which a family of variables is indexed by a list of categories, ideally it should suffice to give just one definition, in which any particular category can be substituted. We show how ths can be achieved in our language.

Let x be a list of product categories, like *shoes, pants, shirts,* etcetera. Let, for each category $d \in x$, v_d be the variable *turnover generated by the sales of* d. Thus we consider the variables *turnover generated by the sales of shoes*, *turnover generated by the sales of pants*, etcetera. Since each of these variables is assumed to be measured on a business, and to record a quantity of money, we let p be the object type *business*, q be the value type *quantity of money* and we thus assume that $\mathsf{v}_d :: [\mathsf{p} \to \mathsf{q}]$ for each d.

Now consider the type r of product categories (i.e., we assume that $\mathcal{I}(\mathsf{r}) = x$). We let each product category be a constant w_d of type $[1 \to \mathsf{r}]$. Finally, we let v be an element of type $[\mathsf{r} \to [\mathsf{p} \to \mathsf{q}]]$. The intuitive meaning of v is: given a value of type r, it returns an element of type $[\mathsf{p} \to \mathsf{q}]$. Hence it behaves as an r-indexed family of variables of type $[\mathsf{p} \to \mathsf{q}]$. Now let the intuitive meaning of v be *turnover generated by the sales of [...]*, where *[...]* indicates the substitution point of a particular product category. It is natural now to let $\mathsf{v}_d = \mathsf{v} \circ \mathsf{w}_d$. Note that the composition makes sense and is well-formed. Note also however that $\mathsf{v} \circ \mathsf{w}_d :: [1 \to [\mathsf{p} \to \mathsf{q}]] \neq [\mathsf{p} \to \mathsf{q}]$. This can be solved by adding to our congruence in Definition 5 the law $[1 \to [\mathsf{p} \to \mathsf{q}]] \cong [\mathsf{p} \to \mathsf{q}]$ (or even: $[1 \to \mathsf{p}] \cong \mathsf{p}$) which

makes sense because the sets both sides of the equation represent are in a one-to-one correspondence, i.e., they are isomorphic. Note however that Proposition 4(v) no longer holds in that case, but we claim that a weaker version involving such an isomorphism does.

Next, we study the way subset inclusion can be used to organize variables according to the object types they best apply to.

Example 3. Let p be the object type *business*, let q be the value type of *economic activities*, like *agriculture, mining, construction*, etcetera, and let $\mathsf{v} : [\mathsf{p} \to \mathsf{q}]$ be the variable *main economic activity*. We assume that each economic activity is reflected as a constant $[1 \to \mathsf{q}]$, so we have, e.g., $\mathsf{agr}, \mathsf{min}, \mathsf{con} :: [1 \to \mathsf{q}]$. The object type *farm* can now be defined as to contain those businesses whose main activity is agriculture, i.e., formally as $\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{agr1}(\mathsf{p}))$. The fact that each farm is a business is reflected by $\boldsymbol{\iota}(\mathsf{v}{\sim}\mathsf{agr1}(\mathsf{p})) :: [\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{agr1}(\mathsf{p})) \to \mathsf{p}]$. The variable w of *number of livestock* applies to farms and not so much to the full object type of *business*, so it is natural to treat it as a variable of type $[\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{agr1}(\mathsf{p})) \to \mathsf{r}]$ where we let r be a value type corresponding to categories including [0..99], [100..999] and [1000..4999], say. Note that we now are in a position to define the object type *small farm* based on the number of livestock, as, e.g., $\boldsymbol{\sigma}(\mathsf{w}{\sim}[0..99]1(\boldsymbol{\sigma}(\mathsf{v}{\sim}\mathsf{agr1}(\mathsf{p}))))$, where $[0..99] :: [1 \to \mathsf{r}]$. Note that in the formal expression of *small farm*, all the necessary components to understand the object type are present: from right to left it reads that a *small farm* is a *business*, whose *main activity* is *agriculture* and whose *number of livestock* is in *[0..99]*. Next we can define additional variables on the object type *small farm* and give further specializations. We leave this to the reader.

Finally, we show how inclusion can play a role in combining two datasets that both have a variable suitable for matching.

Example 4. Let d_1 be a dataset containing two variables: one is *age of a person*, denoted by $\mathsf{v} :: [\mathsf{p} \to \mathsf{r}]$, and the other is *income of a person in 2015*, denoted by $\mathsf{w}_1 :: [\mathsf{p} \to \mathsf{q}]$. A second dataset d_2 contains *gender of a person*, denoted by $\mathsf{u} :: [\mathsf{p} \to \mathsf{o}]$, and *income of a person in 2016*, denoted by $\mathsf{w}_2 :: [\mathsf{p} \to \mathsf{q}]$. So formally, we let $\mathsf{d}_1 = \langle \mathsf{v}, \mathsf{w}_1 \rangle$ and $\mathsf{d}_2 = \langle \mathsf{u}, \mathsf{w}_2 \rangle$; note that both expressions make sense. Suppose we want to construct a third dataset with variables *age*, *gender* and *income* for those persons whose income in 2015 equals that of 2016. In terms of the given datasets, the expression for the required dataset would read

$$\langle \boldsymbol{\pi}_1\mathsf{d}_1, \boldsymbol{\pi}_1\mathsf{d}_2, \boldsymbol{\pi}_2\mathsf{d}_2 \rangle \boldsymbol{\iota}(\boldsymbol{\pi}_2\mathsf{d}_1{\sim}\boldsymbol{\pi}_2\mathsf{d}_2),$$

which, by law (2') is congruent to

$$\langle \mathsf{v}, \mathsf{u}, \mathsf{w}_2 \rangle \boldsymbol{\iota}(\mathsf{w}_1{\sim}\mathsf{w}_2),$$

which, in turn, is congruent to

$$\langle \mathsf{v}, \mathsf{u}, \mathsf{w}_1 \rangle \boldsymbol{\iota}(\mathsf{w}_1{\sim}\mathsf{w}_2),$$

by laws (3') and (8).

8 Conclusion

In this article we have defined a typed formal language for structurally modeling statistical data. The language includes a natural congruence relation, which provides a mechanism for identifying models of statistical data that are synonymous. We have given the language a sound compositional sets-and-functions semantics and we have proven some natural and desired properties of the language.

Technically, the main contribution of the article is the construction of a congruence relation in the scope of a typed language, in which types depend on 'values', or on elements as they are called here. Incorporating dependent types in a language has as a consequence that semantic techniques such as equational logic [16] don't work anymore. To make it work still, a particular closure operation needs to be constructed.

From a statistical perspective, the main contribution is the introduction of a notion of subtyping that is constructive, in contrast to similar notions from the UML with its generalization-specialization arrow, or from the Resource Description Framework (RDF) [11] based languages such as the Web Ontology Language (OWL) [17] or RDF Schema [4], with its notion of *subClassOf*. We mean by constructive that our notion of subset inclusion incorporates the conditions for the inclusion, in contrast to the other notions. This means that with UML, OWL or any such language we know of, while we can express that a *man* is a *person*, we cannot express that this is the case because of a property called *gender*. We feel that this is an important and natural addition for use within the statistical process, because of the relationship between categorical variables (such as *gender*) and the subclasses they define (viz. *men* and *women*). Moreover, the conditions for subclasses give us the mechanisms for deciding, e.g., given an arbitrary *person*, whether or not he (she) is a *man*.

In a theoretical sense we think of subset inclusion as an instance of the so-called axiom of comprehension from set theory [6]. In its most general form it states that *for any condition on x there exists a set which contains exactly those elements x which fulfill this condition* (see [6, p. 31]). The fact that the axiom of comprehension[6] is indeed a basic axiom from set theory, and is independent from the other axioms, strengthens our belief that subset inclusion is also basic and expressive. Thus we view our construct as the proper translation of the axiom of comprehension to the vocabulary of variables and data sets used in statistics: variables and data sets give us the means to express the proper conditions meant above.

A question left untouched in this article is whether or not it is decidable, given two terms v and w, whether or not $\mathsf{v} \cong \mathsf{w}$. It is crucial that decidability is established, for instance because the typing relation depends on it, cf. Proposition 3(iii): if, for instance, we want to compose two elements, in general this means that we need to make sure that the domain of the one is congruent with the codomain of the

[6] or rather: the axiom schema of subsets, which according to [6], is left of the general axiom of comprehension within Zermelo-Fraenkel set theory.

other. At this moment, we don't know whether $\cong$ is decidable, but we conjecture that it is.

One of the usual means for establishing decidability of a congruence defined on terms (i.e., to decide so-called word problems in an algebra), is to try to define a term rewriting system [2, 12], in which terms are rewritten according to equations (such as the ones defined in Definition 5) that are given a rewriting direction: either left-to-right, or right-to-left. Terms that cannot be rewritten any further are called normal forms; the rewriting mechanism thus turns the problem of deciding $\mathsf{v} \cong \mathsf{w}$ into checking whether or not the normal forms corresponding to v and w are equal or not. For this to work, the rewriting system must be (strongly) normalizing and confluent: every sequence of rewrite steps must eventually terminate with a normal form (i.e., infinite such sequences are not allowed), and, loosely speaking, the application of one rewrite rule does not block the application of another. Usually, *completion* is needed to gain both, in which rewrite rules are added to a system of already established rewrite rules; a procedure that may finish successfuly or unsuccessfully, or run forever (i.e., completion is semidecidable). At the moment we are investigating whether a proper rewriting system can be formulated. Special care must be taken to take into account the conditions on some of the congruence laws of Definition 5 involving $\triangleright$ and ::, and the fact that some of the laws are in fact families of laws. This means that we will have an infinite actual number of rewrite rules, but we claim that this system can be reformulated into an equivalent one with a finite number of rules. Finally, because of the typing relation, a suitable nonstandard notion of a typed term rewriting system must be formulated.

During the formulation of the laws of Definition 5, we were surprised to learn that the 'interaction' between subset inclusion and aggregation is confined to two (one of which rather obscure) laws, viz. laws (29) and (30). This means that in general, aggregation and inclusion are hard to interchange: if we select some rows from a dataset and then aggregate, then in general we cannot arrive at the same result doing it the other way around in most cases. This fact, we feel, is crucial in the formulation of metadata models (or normal forms in the sense above for that matter) that claim to incorporate both inclusion and aggregation.

Acknowledgements

The author is grateful to Sander Scholtus for his careful reading of an earlier version of this article, and for his many useful comments.

References

[1] Abramsky, S., Gabbay, Dov M., and Maibaum, T. S. E., editors. *Handbook of Logic in Computer Science (Vol. 4): Semantic Modelling.* Oxford University Press, Inc., New York, NY, USA, 1995.

[2] Baader, Franz and Nipkow, Tobias. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998.

[3] Barr, Michael and Wells, Charles. *Category Theory for Computing Science*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[4] Brickley, D. and Guha, R.V. RDF schema 1.1 (W3C recommendation), 2014.

[5] Davey, Brian A. and Priestley, Hilary A. *Introduction to lattices and order*. Cambridge University Press, Cambridge, 1990.

[6] Fraenkel, A.A., Bar-Hillel, Y., and Levy, A. *Foundations of Set Theory*. Elsevier Science, 1973.

[7] Gelsema, Tjalling. General requirements for the soundness of metadata models. In *Joint UNECE/Eurostat/OECD work session on statistical metadata (METIS)*, 2008.

[8] Gelsema, Tjalling. The organization of information in a statistical office. *Journal of Official Statistics*, 28(3):413–440, 2012.

[9] Goguen, J. A., Thatcher, J. W., Wagner, E. G., and Wright, J. B. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24(1):68–95, 1977. DOI: `10.1145/321992.321997`.

[10] Grätzer, G. *Universal Algebra*. D. Van Nostrand Company, Princeton New Jersey, 1968.

[11] Hayes, P.J. and Patel-Schneider, P.F. RDF 1.1 semantics (W3C recommendation), 2014.

[12] Klop, J.W. and de Vrijer, R.C. *Term Rewriting Systems*. Cambridge University Press, 2003.

[13] Lane, S. M. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1998.

[14] Manca, V., Salibra, A., and Scollo, G. Equational type logic. *Theoretical Compututer Science*, 77(1–2):131–159, 1990. DOI: `10.1016/0304-3975(90)90118-2`.

[15] Martin, J. and Odell, J.J. *Object-Oriented Methods: A Foundation; UML Edition*. Prentice-Hall, Upper Saddle River, New Jersey, 1998.

[16] Meinke, K. and Tucker, J.V. Universal algebra. In Abramsky, S., Gabbay, M., and Maibaum, T., editors, *Handbook of Logic in Computer Science, Vol. I: Background; Mathematical Structures*. Oxford Science Publications, 1992.

[17] Motik, B., Patel-Schneider, P.F., and Grau, B. Cuenca. OWL 2 web ontology language direct semantics (second edition, W3C recommendation), 2012.

[18] Pierce, B.C. *Basic Category Theory for Computer Scientist.* The MIT Press, Cambridge Massachusetts, 1991.

[19] Pierce, B.C. *Types and Programming Languages.* The MIT Press, Cambridge Massachusetts, 2002.

[20] Signore, M., Scanu, M., and Brancato, G. Statistical metadata: a unified approach to management and dissimination. *Journal of Official Statistics*, 31(2):325–347, 2015. DOI: 10.1515/jos-2015-0020.

[21] Thomson, S. *Type Theory and Functional Programming.* Addison-Wesley, 1994.

[22] United Nations Economic Commission for Europe (UNECE). Generic statistical information model (GSIM): Specification, 2013.

[23] van Leeuwen, J. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics.* The MIT Press, Cambridge Massachusetts, 1994.

Received 10th September 2018

Acta Cybernetica **24** (2019) 249–262.

Reconstruction of Rooted Directed Trees*

Dénes Bartha[a]

Abstract

Let T be a rooted directed tree on n vertices, rooted at v. The rooted subtree frequency vector (*RSTF-vector*) of T with root v, denoted by $\mathrm{rstf}(T, v)$ is a vector of length n whose entry at position k is the number of subtrees of T that contain v and have exactly k vertices. In this paper we present an algorithm for reconstructing rooted directed trees from their rooted subtree frequencies (up to isomorphism). We show that there are examples of nonisomorphic pairs of rooted directed trees that are *RSTF-equivalent*, that is they share the same rooted subtree frequency vectors. We have found all such pairs (groups) for small sizes by using exhaustive computer search. We show that infinitely many nonisomorphic *RSTF-equivalent* pairs of trees exist by constructing infinite families of examples.

Keywords: tree reconstruction, subtree size frequencies, rooted directed trees

1 Introduction

Reconstruction of certain combinatorial structures from given partial information plays an important role in several problems such as reconstructibility of strings [5, 3, 1], trees, graphs [8, 7], matrices [9, 4] etc.

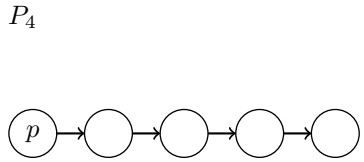
The motivation behind this paper comes from mass spectrometry data analysis. The problem we investigate is the possibility of reconstruction of an unlabeled directed rooted tree with n vertices, given the number of rooted directed subtrees frequencies of size $1, 2, \ldots, n$, which we call the *RSTF-vector*. In [2] the authors investigated the problem of reconstructibility of unlabeled free trees and defined *STF-vector* with the sum of all the *RSTF-vectors* of the subtrees of a given tree. Because there is no reconstruction algorithm of free trees given in the literature, the approach presented in this paper could be the first step towards such an algorithm.

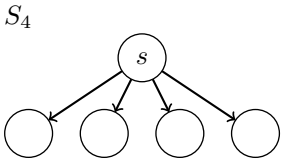
In Section 2 we give the formal definition of the *RSTF-vector*, *RSTF-polynomial*, *well formed representation* and show how to construct them from a given rooted

*This submission is for the special issue of CSCS 2018. Talent Management in Autonomous Vehicle Control Technologies – The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

[a]Eötvös Loránd University, E-mail: denesb@gmail.com

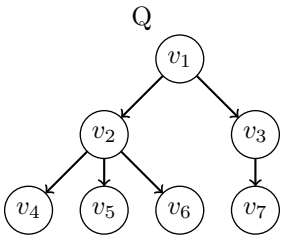
DOI: 10.14232/actacyb.24.2.2019.5





$\text{rstf}(P_4, p) = [1, 1, 1, 1, 1]$
$r(P_5, p) = 1 + x + x^2 + x^3 + x^4$

$\text{rstf}(S_4, s) = [1, 4, 6, 4, 1]$
$r(S_4, s) = 1 + 4x + 6x^2 + 4x^3 + x^4$



$\text{rstf}(Q, v_1) = [1, 2, 5, 7, 7, 4, 1]$
$r(Q, v_1) = 1 + 2x + 5x^2 + 7x^3 + 7x^4 + 4x^5 + x^6$

Figure 1: P_4 denotes a path of length 4 rooted at p, S_4 a star with 4 leaves rooted at v_2, and a more complex tree Q on 7 nodes rooted at v_1. The corresponding *RSTF-vectors* and *RSTF-polynomials* are given below the trees.

directed tree. In section 3 we introduce the algorithm *RRDT* that can reconstruct the rooted directed tree corresponding to a given polynomial. In section 4 we show some results on *RSTF-equivalent* trees. In the Conclusion section we propose new research directions.

The main problem we investigate is the method of reconstructing an unlabeled rooted directed tree from its rooted subtree frequencies [2]. The motivation of the problem comes from mass spectrometry data analysis, as the *RSTF-vector* models frequency data from mass spectrometry. Although unique reconstruction from this vector is not always possible, we still find the mathematical and algorithmic aspects of the problem worth investigating. Also, as a practical application, a molecule database search filter might be created using RSTF indexing, but this is outside the scope of the present paper and could be the topic of future research.

2 Basic definitions

In the paper x is used for the variable of univariate polynomials denoted by $f, g, \ldots$ Unless otherwise stated, polynomials have integer coefficients. The letter n usually denotes the number of nodes of an unlabeled rooted directed tree. Trees are denoted

by capital letters $P, Q, R, S, \ldots$

Definition 1. *Let $T = (V, E)$ be a rooted directed tree on $n (n \geq 1)$ vertices, rooted at vertex $v \in V$. The vector* $\mathrm{rstf}(T, v) = [r_1, \ldots, r_n]$ *is called the rooted subtree frequency vector (RSTF for short) of T with root v, where each r_i shows the number of those i-sized subtrees of T that contain v.*

We can represent *RSTF-vectors* with polynomials by choosing the entries of the vectors as the appropriate coefficients of the polynomial.

Definition 2. *Let T be a rooted directed tree, v the root of T, with* $\mathrm{rstf}(T, v) = [r_1, r_2, \ldots, r_n]$. *The* RSTF-polynomial *of T with root v, denoted by $r(T, v)$ is defined by $r(T, v) = r_1 + r_2 x + r_3 x^2 + \cdots + r_n x^{n-1}$.*

Figure 1 shows three examples on *RSTF-vectors* and *polynomials*. The reason why we use *RSTF-polynomials* instead of vectors is that we can easily calculate the *RSTF-polynomial* from a given rooted directed tree graph structure as shown in Lemma 1 [2].

Lemma 1. *Given a tree T with root v, one can calculate the RSTF-polynomial in $O(n^2)$ time using the following recursive formula:*

$$\mathrm{rstf}(T, v) = \prod_{i=1}^{k} (1 + x \cdot \mathrm{rstf}(T_i, v_i)) \,,$$

where k is the number of children of v, which are denoted by v_i, and T_i is the subtree rooted at v_i.

To illustrate the use of this lemma, we compute the *RSTF-polynomial* of tree Q given in Figure 1, applying the recursive approach step by step (Q_i denotes the subtree rooted at v_i, $i = 2, \ldots, 7$):

$$\begin{aligned}
\mathrm{rstf}(Q, v_1) &= (1 + x \cdot \mathrm{rstf}(Q_2, v_2)) \cdot (1 + x \cdot \mathrm{rstf}(Q_3, v_3)) \\
\mathrm{rstf}(Q_2, v_2) &= (1 + x \cdot \mathrm{rstf}(Q_4, v_4)) \cdot (1 + x \cdot \mathrm{rstf}(Q_5, v_5)) \cdot (1 + x \cdot \mathrm{rstf}(Q_6, v_6)) \\
\mathrm{rstf}(Q_3, v_3) &= (1 + x \cdot \mathrm{rstf}(Q, v_7)) \\
\mathrm{rstf}(Q_4, v_4) &= \mathrm{rstf}(Q_5, v_5) = \mathrm{rstf}(Q_6, v_6) = \mathrm{rstf}(Q_7, v_7) = 1
\end{aligned}$$

If we substitute back to $\mathrm{rstf}(Q, v_1)$, and expand the product, the resulted polynomial's coefficient sequence gives the *RSTF-vector* of Q:

$$\begin{aligned}
\mathrm{rstf}(Q, v_1) &= (1 + x \cdot ((1 + x \cdot 1) \cdot (1 + x \cdot 1) \cdot (1 + x \cdot 1)) \cdot (1 + x \cdot (1 + x \cdot 1))) \\
&= \mathbf{1} + \mathbf{2}x + \mathbf{5}x^2 + \mathbf{7}x^3 + \mathbf{7}x^4 + \mathbf{4}x^5 + \mathbf{1}x^6
\end{aligned}$$

A polynomial may have many different representations, but in the case of *RSTF-polynomials*, there is a specific representation from which the tree structure corresponding to the polynomial is easy to determine (as in the above form of $\mathrm{rstf}(Q, v_1)$. We therefore introduce the following formal definition.

Definition 3. *We call a representation of a polynomial f well-formed, if it is either of the form*

- $f = 1$, *or*
- $f = (1+x\cdot f_1)\cdot(1+x\cdot f_2)\cdots(1+x\cdot f_k)$ *with* $k \geq 1$ *where the* f_j $(k = 1,\ldots,k)$ *are themselves polynomials in well-formed representation.*

Every polynomial f with a well-formed representation is an *RSTF-polynomial*, and every *RSTF-polynomial* has a well-formed representation. Using the notations in the definition, the connection is that the root has k children and the subtrees rooted in these children have *RSTF-polynomials* f_j for $j = 1,\ldots,k$. Note that well-formed representations consist only of 1, +, x, $\cdot$ and () symbols, and they can be generated by a context-free grammar.

A necessary but not sufficient condition for a polynomial f to have a well-formed representation is that it can be written as $f = (1+xf_1)\cdot(1+xf_2)\cdots(1+xf_k)$ with polynomials f_j that have constant term 1 (but not necessarily RSTF-polynomials themselves). We will use this as a filter in our algorithms later. For later use, we define the concept of *RSTF-candidate factor* and *RSTF-candidate representation.*

Definition 4. *We call a polynomial RSTF-candidate factor if both the constant and linear coefficients are equal to 1. We call a representation of polynomial f RSTF-candidate representation if f is written as a product of RSTF-candidate factors, i.e. as $f = (1+xf_1)\cdot(1+xf_2)\cdots(1+xf_k)$ where all polynomials f_j have constant term* 1.

As explained later, the algorithms for finding well-formed representations (or otherwise put, corresponding trees) for a polynomial f will operate by first finding all *RSTF-candidate representations* and then recursively checking whether the polynomials f_j are RSTF-polynomials or not, and if they are, giving all their well-formed representations.

3 Methods

3.1 Reconstruction algorithm

Our main goal is to construct an algorithm that has a polynomial f as input and all trees having f as *RSTF-polynomial* as output. With the help of Lemma 1 we can construct such an algorithm. As discussed in the previous section, for finding the tree structure it is sufficient to give the well-formed representation of the polynomial.

The main idea is to factorize the input polynomial into irreducible factors and then group the factors so that this grouping yields a *well-formed* representation (using recursive calls in the process).

Let a_1 be the linear coefficient of f and let p_i denote the distinct irreducible factors of f, with exponent k_i. Then the irreducible factorization and the well-formed representation (necessarily having a_1 factors by matching the linear terms) are both equal to f:

$$\begin{aligned} f &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \\ &= p_1^{k_1} \cdot p_2^{k_2} \cdots p_m^{k_m} \\ &= (1 + x \cdot f_1) \cdot (1 + x \cdot f_2) \cdots (1 + x \cdot f_{a_1}) \end{aligned} \tag{1}$$

Each factor in the well-formed representation is the product of some irreducible factors. We call a partition of the multiset of irreducible factors a *proper grouping* of irreducible factors if the product of polynomials within the groups is always of the form $1 + xf_j$ where f_j is an RSTF-polynomial.

Once we have a well-formed representation, the tree structure can be given quickly:

Lemma 2. *From a well formed RSTF-polynomial on degree $n-1$ we can reconstruct a corresponding tree in n steps.*

Proof. Let f be a *well formed RSTF-polynomial* corresponding to a tree. Since $deg(f) \geq 0$ we can assume that the tree has at least one node that we create in advance. Then using Lemma 1 we have to count the number of outer blocks $(\ldots) \cdots (\ldots)$ and create as many new nodes (children) on the next level that we connect with the previous parent node with a directed edge (the edge points to the children). We build up the tree using this simple rule for each block recursively. Because there are exactly n pair of parentheses jumbled in f, this process takes n steps. □

The pseudocode of the *RRDT* algorithm can be seen in Algorithm 1. Figure 3.1 shows an example on how it works.

For any given polynomial the function gives back the corresponding well-formed polynomial (or polynomials) if one exists, otherwise returns $\downarrow$. First it checks the base cases. Note that the constant term and the linear coefficient must be equal to 1 (see Lemma 1). The next step is to check whether the solution can be found in the dictionary (*known*) by using dynamic programming approach (memoization). The upcoming part consists of two main cases: when the linear coefficient of the given polynomial (denoted by $f[x]$) is 1 and when it is greater than 1. Note that because of the well-formed property equation (1) it cannot be 0 or negative.

When $f[x] = 1$, we have to call the function recursively for $\frac{f-1}{x}$ because if there is a solution, then f has the form: $1 + x \cdot (\ldots)$. In this case we have to store $(1 + x \cdot result)$ in the dictionary.

Otherwise if $f[x] > 1$, we perform polynomial factorization that can be computed efficiently (polynomial time) using LLL [6] or other similar methods. The factorization gives the prime factors with the appropriate powers in the form of $f = p_1^{k_1} \cdots p_m^{k_m} \cdot q_1^{k_{m+1}} \cdots q_s^{k_{m+s}}$ (where p_i and q_j are prime factors). We can observe that among the factors there will be *RSTF-polynomials* p_i that can be represented by well-formed polynomials according to equation (1) and the remaining non-*RSTF-polynomials* q_j have different form (e.g. have a constant term different from 1).

Algorithm 1 Reconstruct well-formed RSTF-polynomial

$known \leftarrow \{\}$ ▷ dictionary for the known polynomials
procedure RRDT($f = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$)
  if $f = 1$ **then**
    return 1 ▷ Base cases
  else if $a_0 \neq 1 \vee a_n \neq 1$ **then**
    return $\downarrow$
  else if $f \in known$ **then**
    return $known[f]$
  else if $a_1 = 1$ **then** ▷ If the linear coefficient is 1
    $rp \leftarrow \text{RRDT}(\frac{f-1}{x})$;
    if $rp = \downarrow$ **then**
      $known[f] \leftarrow \downarrow$
    else
      $known[f] \leftarrow (1 + x \cdot rp)$
    end if
  else ▷ If the linear coefficient is greater than 1
    $f = r_1^{k_1} \cdots r_t^{k_t}$ ▷ Factorization step
    if $t < a_1$ **then** ▷ If there are less factors than $a_1 \rightarrow$ no solution
      $known[f] \leftarrow \downarrow$
    else
      $f = p_1^{k_1} \cdots p_m^{k_m} \cdot q_1^{k_{m+1}} \cdots q_s^{k_{m+s}}$ ▷ p_i: *RSTF-polynomials*
      ▷ q_j: non-*RSTF-polynomials*
      ▷ Find the proper groupings: $g_1, \ldots, g_{a_1}$
      if $\exists f = g_1 \cdots g_{a_1}, \forall i \in [1, a_1] : \text{RRDT}(g_i) \neq \downarrow$ **then**
        $rp \leftarrow \prod_{i=1}^{a_1} (1 + x \cdot \text{RRDT}(\frac{g_i - 1}{x}))$
        $known[f] \leftarrow rp$
      else
        $known[f] \leftarrow \downarrow$
      end if
    end if
  end if
  return $known[f]$
end procedure

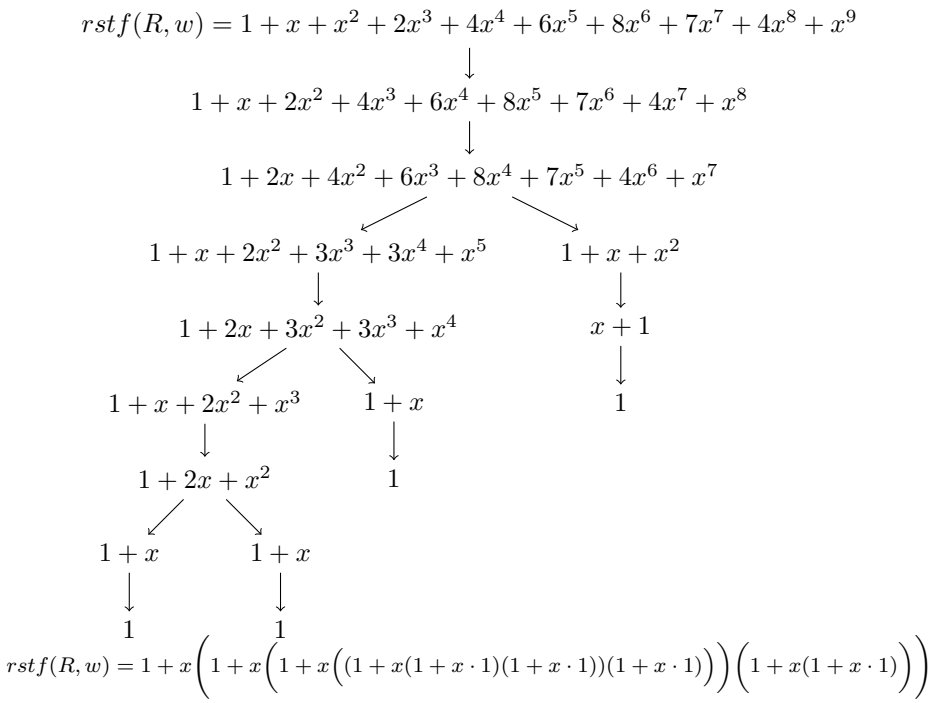


Figure 2: Given a polynomial $1 + x + x^2 + 2x^3 + 4x^4 + 6x^5 + 8x^6 + 7x^7 + 4x^8 + x^9$. Each node in the graph (except the root w) is constructed by and connected with the appropriate parent node.

We need to find proper groupings of the prime factors $f = g_1 \cdot g_2 \cdots g_{a_1}$, where each g_i is already well-formed. Note that the number of such groups is exactly $f[x] = a_1$. This step is nontrivial and needs further explanation how it can be done reasonably fast, therefore we devote the following subsection to this subroutine.

3.2 Find the proper groupings

The naïve approach is to try all the possible combinations of the prime factors and filter out the proper settings (each group represents a *rooted, directed tree*). In most cases (when we don't have many different kind of prime factors) this approach - generating all the a_1-sized partitions of a multiset - could work. But note that this is even worse than finding all the permutations of a multiset (with repetitions allowed) because we also have to distribute parentheses.

Fortunately we can give a better method to solve this problem. By the well-formed property, in each group of a proper grouping, the linear coefficient of the product must be 1. This is seen by expanding the product of all the members of the group: $1 + x + b_1x^2 + \ldots + b_{l-1}x^{l-1} + x^l$ (for some degree l).

Recall that *RSTF-candidate representations* are exactly the ones having this property of the linear term. So we will look at all *RSTF-candidate representations*, and then we have to recursively check whether the *RSTF-candidate factors* are indeed *RSTF-polynomials* or not, and if they are, function RRDT gives all their well-formed representations recursively.

A grouping of irreducible factors that gives an RSTF-candidate representation is easy to verify: we only need to sum the linear coefficients. We call a partition of a multiset of *integers* a proper integer grouping if the sum in every group is exactly 1.

Note that the prime factorization could give back non-*RSTF-polynomials* where the constant term is not equal to one (negative, 0 or greater than 1). Hence we first create a multiset of the linear coefficients of the *prime polynomials* $p_i[x]$, $q_j[x]$. We then find all proper integer groupings where the sum of each group is 1. In Algorithm 2, calling FindProperIntegerGroupings$(A, \emptyset, \emptyset)$ for some multiset of integers A will output all such proper integer groupings. Note that this uses a *DFS* approach.

Algorithm 2 Finding the proper grouping of an integer multiset

```
1: procedure FindProperIntegerGroupings(A, G, group)
2:     if ∑_{g_i ∈ group} g_i > 1 or ∑_{a_i ∈ A} a_i < 1 then
3:         return
4:     end if
5:     if ∑_{g_i ∈ group} g_i = 1 then
6:         G ← G ∪ {group}
7:         group ← ∅
8:         if A = ∅ then
9:             output G
10:        end if
11:    end if
12:    for ∀a ∈ A do
13:        FindProperIntegerGroupings(A \ {a}, G, group ∪ {a})
14:    end for
15: end procedure
```

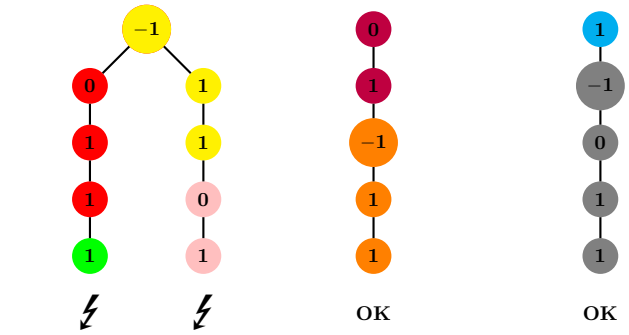
Note that FindProperIntegerGroupings might output the same partition multiple times. To avoid this we introduce the following ideas.

We define a $\preceq$ relation on the subsets (multisets) of a finite set $A \subset \mathbb{Z}$ in the following way: $\forall x, y \subseteq A$: (where x, y are multisets) $x \preceq y \iff |x| < |y|$ *or* $(|x| = |y|$ *and* $[x] \leq [y])$, where $x = \{x_1 \cdot d_1, x_2 \cdot d_2, \ldots, x_n \cdot d_n\}$, $x_1 < x_2 < \cdots < x_n$, $[x] = [\underbrace{x_1, \ldots, x_1}_{d_1}, \underbrace{x_2, \ldots, x_2}_{d_2}, \ldots \underbrace{x_n, \ldots, x_n}_{d_n}]$ here $\leq$ represents the lexicographic relation. Now we can extend function FindProperIntegerGroupings with the following things:

- In line 13 take the elements of A in *monotonically increasing order.*

- Add a new line between line 5 and 6: "$if\ \neg(y \preceq group)\ then\ return$", where y denotes the last group that we have added to G.

The following figure shows an example on how to find the groupings of the multiset $A = \{-1, 0, 1, 1, 1\}$ (the colours denote different groups).



Solutions: $\Big\{\big\{\{-1,0,1,1\},\{1\}\big\},\big\{\{0,1\},\{-1,1,1\}\big\}\Big\}$

After this step another problem arises: there could be more samples of each type of polynomials, where the type corresponds to the linear coefficient. Consider the following example:

$$1 + 2x + 4x^2 + 8x^3 + 12x^4 + 15x^5 + 16x^6 + 15x^7 + 11x^8 + 5x^9 + x^{10}$$
$$= \underbrace{(1 + \mathbf{0}\cdot\mathbf{x} + x^2 + 2x^3 + x^4)}_{f_1^{(0)}} \cdot \underbrace{(1 + \mathbf{0}\cdot\mathbf{x} + x^2 + x^3)}_{f_2^{(0)}} \cdot \underbrace{(1 + \mathbf{1}\cdot\mathbf{x} + x^2)}_{f_3^{(1)}} \cdot \underbrace{(1 + \mathbf{1}\cdot\mathbf{x})}_{f_4^{(1)}}$$

Here multiset $A' = \{0, 0, 1, 1\}$ contains the linear coefficients and FindProperIntegerGroupings $(A', \emptyset, \emptyset)$ gives the proper integer groupings:

$$\Big\{\underbrace{\big\{\{0,1\},\{0,1\}\big\}}_{g_1},\underbrace{\big\{\{1\},\{0,0,1\}\big\}}_{g_2}\Big\}$$

But there are different *RSTF-candidate factors*: $\mathbf{0}: f_1^{(0)}, f_2^{(0)}$, $\mathbf{1}: f_3^{(1)}, f_4^{(1)}$ and the question is how to combine them properly:

$$\left.\begin{aligned} g_1 &= \left(f_1^{(0)} \cdot f_3^{(1)}\right) \cdot \left(f_2^{(0)} \cdot f_4^{(1)}\right) \\ g_1 &= \left(f_1^{(0)} \cdot f_4^{(1)}\right) \cdot \left(f_2^{(0)} \cdot f_3^{(1)}\right) \end{aligned}\right\} \text{Is it valid?}$$

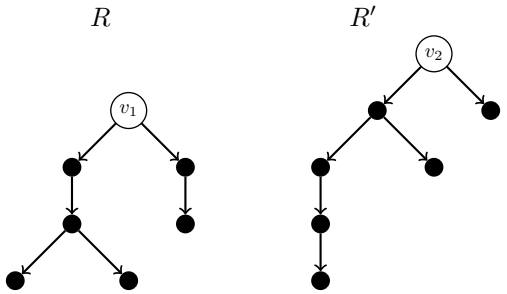


Figure 3: R and R' are two nonisomorphic rooted directed *RSTF-equivalent* trees

$$\left.\begin{aligned} g_2 &= \left(f_3^{(1)}\right)\cdot\left(f_1^{(0)}\cdot f_2^{(0)}\cdot f_4^{(1)}\right)\\ g_2 &= \left(f_4^{(1)}\right)\cdot\left(f_1^{(0)}\cdot f_2^{(0)}\cdot f_3^{(1)}\right)\end{aligned}\right\}\text{ Is it valid?}$$

Fortunately it is not so hard to get the proper settings from the possibilities if we use the above presented function *RRDT*. If one *RSTF-candidate factor* does not represent a valid *rooted directed tree*, we don't need to check the remaining factors. In this case we have to carry on and check the next possible *RSTF-candidate representation* until we find a proper solution.

Function *RRDT* reduces the degree of the polynomial by 1 when $f[x] = 1$ (or returns with a saved result). When $f[x] > 1$ the factorization step takes polynomial time. Hence the step of finding the proper grouping dominates the function where artificial examples could be given that takes exponential running time. However in practice it works fine for bigger trees on 500-1000 nodes as well and finds the solutions in a few seconds.

An implementation of the RRDT-algorithm written in sage, python can be found at `https://github.com/denesbartha/RRDT`.

4 Isomorphism and reconstructibility results

Algorithm 1 is able to reconstruct rooted directed trees up to isomorphism. There are several cases when the given polynomials determine uniquely the trees. Typical examples (see Figure 1) are P_m - path of length m (coefficients of the corresponding polynomial: $1, 1, \ldots 1$) and S_k - star with k leaves (coefficients of the corresponding polynomial: $\binom{k}{0}, \binom{k}{1}, \ldots \binom{k}{k}$) [2]. Not surprisingly there are cases when a given input polynomial represents multiple rooted directed trees. Figure 3 shows two nonisomorphic rooted directed trees that share the same *RSTF-polynomial*.

Definition 5. *We call two nonisomorphic rooted directed trees RSTF-equivalent if they share the same* RSTF-polynomial.

The following equation shows the *well formed RSTF-polynomials* of R and R' trees.

$$
\begin{aligned}
&\mathrm{rstf}(R, v_1) = \mathrm{rstf}(R', v_2) \\
&= x^6 + 3x^5 + 4x^4 + 4x^3 + 3x^2 + 2x + 1 \\
&= (x^3 + x^2 + 1) \cdot (x^2 + x + 1) \cdot (x + 1) \\
&= \Big(1 + x \cdot \Big(1 + x \cdot \Big(\big(1 + x \cdot 1\big) \cdot \big(1 + x \cdot 1\big)\Big)\Big)\Big) \cdot \Big(1 + x \cdot \big(1 + x \cdot 1\big)\Big) \\
&= \Big(1 + x \cdot \Big(\Big(1 + x \cdot \big(1 + x \cdot (1 + x \cdot 1)\big)\Big) \cdot \Big(1 + x \cdot 1\Big)\Big)\Big) \cdot (1 + x \cdot 1)
\end{aligned}
$$

Lemma 3. *Given two nonisomorphic rooted directed RSTF-equivalent trees T_1 and T_2, that share the same RSTF-polynomial f. If we add a new node respectively to both trees that we connect with the original roots, the resulted T_1', T_2' trees will remain RSTF-equivalent. Their RSTF-polynomial is $1 + x \cdot f$.*

Proof. By using Lemma 1 we can see that joining a new root node to a rooted directed tree Q with RSTF-polynomial g results in a new tree Q' that has RSTF-polynomial $1 + x \cdot g$. Simply applying this rule to the given nonisomorphic rooted directed RSTF-equivalent trees T_1 and T_2 with RSTF-polynomial f, we create two new nonisomorphic rooted directed trees T_1' and T_2' that share the RSTF-polynomial $1 + x \cdot f$ □

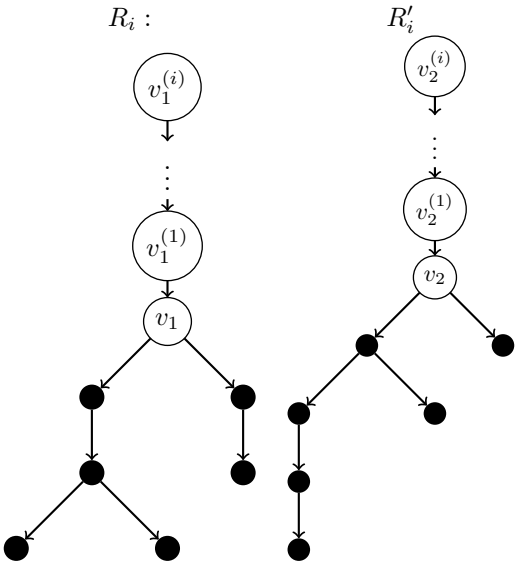


Figure 4: $\mathrm{rstf}(R_i, v_1^{(i)}) = \mathrm{rstf}(R_i', v_2^{(i)}) = 1 + x \cdot (1 + x \cdot (\ldots (1 + x \cdot \mathrm{rstf}(R, v_1)) \ldots)),\ \forall i \in \mathbb{N}^+$

Table 1: second column: $a(n)$ - the number of unlabeled rooted trees with n nodes (`https://oeis.org/A000081`); third column: the number of nonisomorphic equivalence classes; fourth column: the ratio of #equivalence classes to $a(n)$; fifth column: the maximal size equivalence class; sixth column: Shannon entropy of the equivalence classes.

n	**a(n)**	#equivalence classes	$\frac{\mathbf{ec(n)}}{\mathbf{a(n)}}$	Maximal size equivalence class	Entropy
3	2	2	1.0	1	0
4	4	4	1.0	1	0
5	9	9	1.0	1	0
6	20	20	1.0	1	0
7	48	47	0.97917	2	0.14855
8	115	112	0.97391	2	0.178
9	286	274	0.95804	2	0.25943
10	719	679	0.94437	2	0.3231
11	1842	1717	0.93214	3	0.3833
12	4766	4393	0.92174	4	0.42953
13	12486	11374	0.91094	4	0.47557
14	32973	29725	0.9015	5	0.51466
15	87811	78428	0.89315	7	0.54811
16	235381	208431	0.8855	8	0.57819
17	634847	557555	0.87825	11	0.60622
18	1721159	1499739	0.87135	11	0.63245
19	4688676	4054714	0.86479	15	0.65711
20	12826228	11011259	0.8585	16	0.68046

Theorem 1. *There are infinitely many RSTF-equivalent pairs of trees exist.*

Proof. It is enough if we find only one RSTF-equivalent pair of rooted directed trees. By joining arbitrary many new nodes to their roots respectively (Lemma 3) we always get new nonisomorphic rooted directed *RSTF-equivalent* trees. For example we can alter the given pair of directed trees rooted at v_1, v_2 in Figure 4 by joining new roots to them respectively arbitrary many times. This creates new nonisomorphic rooted directed *RSTF-equivalent* pair of trees. □

RSTF-equivalency forms an equivalence relation where the classes are the sets of nonisomorphic rooted directed trees with n nodes. Using exhaustive computer search we have found all the equivalence classes up to $n = 20$. Table 1 summarizes the results. Here we applied the Shannon entropy of the equivalence classes s.t. for a fixed tree size n that has $a(n)$ number of nonisomorphic rooted directed trees with *RSTF-equivalent* classes of $C(n) = \{c_1, \dots c_m\}$, $m \leq n$, $H_0(C(n)) = -\sum \log_2(|c_i|/m)|c_i|/m$. Up to $n = 6$, $H_0(C(n)) = 0$ which means that every

equivalence class contains only one element (in other words there are no *RSTF-equivalent* pairs). For $n > 6$ sizes the entropy rises.

Note that the maximum number of equivalence classes $ec(n)$ cannot exceed the number of nonisomorphic rooted directed trees $a(n)$, hence $\frac{ec(n)}{a(n)} \leq 1$. Also because there are infinite nonisomorphic rooted directed *RSTF-equivalent* tree classes exist (Lemma 1), $0 < \frac{ec(n)}{a(n)} < 1$, for $n \geq 7$. Our conjecture is that $lim_{n\to\infty} \frac{ec(n)}{a(n)} = 0$.

5 Conclusion and future work

In this paper we gave a concrete reconstruction algorithm *RRDT* for rooted directed trees. The main future goal is to extend these results for free trees / simple graphs that could be used in bioinformatics or spectrometry data analysis. We also plan to analyze the time complexity of the algorithm formally.

We were using only univariate polynomials and unlabeled trees. We aim to extend the above presented approach using multivariate polynomials representing labeled rooted directed trees.

Although it seems hard, in theory the factorization step of the reconstruction algorithm could be modified s.t. it would produce correct groupings of a given polynomial that represents a rooted directed tree.

I would also like to thank the anonymous referee for the valuable comments and suggestions.

References

[1] Acharya, Jayadev, Das, Hirakendu, Milenkovic, Olgica, Orlitsky, Alon, and Pan, Shengjun. String reconstruction from substring compositions. *SIAM Journal on Discrete Mathematics*, 29(3):1340–1371, 2015. DOI: 10.1137/140962486.

[2] Bartha, Dénes and Burcsi, Péter. Reconstructibility of trees from subtree size frequencies. *Studia Universitatis Babes-Bolyai, Mathematica*, 59(4):435–442, 2014.

[3] Dudik, Miroslav and Schulman, Leonard J. Reconstruction from subsequences. *Journal of Combinatorial Theory, Series A*, 103(2):337–348, 2003. DOI: 10.1016/s0097-3165(03)00103-1.

[4] Kós, Géza, Ligeti, Péter, and Sziklai, Péter. Reconstruction of matrices from submatrices. *Mathematics of Computation*, 778:1733–1747, 2009. DOI: 10.1090/s0025-5718-09-02210-8.

[5] Krasikov, I. and Roditty, Y. On a reconstruction problem for sequences. *Journal of Combinatorial Theory, Series A*, 77(2):344–348, 1997. DOI: 10.1006/jcta.1997.2732.

[6] Lenstra, A. K., Lenstra, H. W., and Lovász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. DOI: 10.1007/bf01457454.

[7] Manvel, Bennet. Reconstruction of trees. *Canadian Journal of Mathematics*, 22(1):55–60, 1970. DOI: 10.4153/CJM-1970-007-4.

[8] Manvel, Bennet. On reconstructing graphs from their sets of subgraphs. *Journal of Combinatorial Theory, Series B*, 21(2):156–165, 1976. DOI: 10.1016/0095-8956(76)90056-3.

[9] Manvel, Bennet and Stockmeyer, Paul K. On reconstruction of matrices. *Mathematics Magazine*, 44(4):218–221, 1971. DOI: 10.2307/2689082.

Received 7th September 2018

Contents

ISSN 0324—721 X (Print)
ISSN 2676—993 X (Online)

Editor-in-Chief: Tibor Csendes